

**VISVESVARAYA TECHNOLOGICAL UNIVERSITY**  
**Jnana Sangama, Belagavi-590010**



**DBMS MINI PROJECT REPORT**  
**ON**

**“STUDENT MANAGEMENT SYSTEM ”**

Submitted in partial fulfillment for the requirements for the fifth semester

**BACHELOR OF ENGINEERING**  
**IN**  
**INFORMATION SCIENCE AND ENGINEERING**

For the Academic Year 2021-2022

Submitted by:

**DHARMA PRASAD**

**1MV20IS400**

**MANOJ.V**

**1MV20IS401**

**SUMANTH .B**

**1MV20IS403**

**SUNIL KUMAR G.P**

**1MV20IS404**

**VIJAY RAJ SINGH**

**1MV20IS405**

Under the guidance of:

**VIJAY KUMAR**

Professor, Department of ISE

Sir M. Visvesvaraya Institute of Technology, Bengaluru



**DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING**  
**SIR M. VISVESVARAYA INSTITUTE OF TECHNOLOGY**  
**HUNASAMARANAHALLI, BENGALURU-562157**

**SIR M. VISVESVARAYA INSTITUTE OF TECHNOLOGY**

Krishnadevaraya Nagar, International Airport Road,

Hunasamaranahalli, Bengaluru – 562157

**DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING**



**CERTIFICATE**

It is certified that the DBMS Mini Project work entitled " STUDENT MANAGEMENT SYSTEM (S.M.S)" is carried out by **DHARMA PRASAD(1MV20IS400),MANOJ.V(1MV20IS401),SUMANTH.B(1MV20IS403),SUNILKUMAR 1MV20IS404),,VIJAY RAJ SINGH(1MV20IS405)** bonafide students of Sir M Visvesvaraya Institute of Technology in partial fulfillment for the 5th semester for the award of the Degree of Bachelor of Engineering in Computer Science and Engineering of the Visvesvaraya Technological University, Belagavi during the academic year 2021-2022. It is certified that all corrections and suggestions indicated for Internal Assessment have been incorporated in the report deposited in the department library. The project report has been approved as it satisfies the academic requirements in respect of project work prescribed for the course of Bachelor of Engineering.

Name & Signature  
Of Guide

Name & Signature  
of HOD

Name & Signature  
of Principal

-----  
**Vijay kumar**  
Prof & Internal Guide  
Dept.of.ISE,sir MVIT  
Bengaluru-562157

-----  
**DR.P.VIJAYKARTHIK**  
Prof & Internal Guide  
Dept.of.ISE,sir MVIT  
Bengaluru-562157

-----  
**Dr. V.R. Manjunath**  
Principal,  
Sir MVIT  
Bengaluru – 562157

External Examination:

Name of Examiner

1)

2)

Signature with Date

## **DECLARATION**

We hereby declare that the entire project work embodied in this dissertation has been carried out by us and no part has been submitted for any degree or diploma of any institution previously.

Place: Bengaluru

Date:

Signature of Students:

-----  
DHARMA PRASAD  
(1MV20IS405)

-----  
MANOJ.V  
(1MV20IS401)

-----  
SUMANTH.B  
(1MV20IS403)

-----  
SUNIL KUMAR G.P  
(1MV20IS404)

-----  
VIJAY RAJ SINGH  
(1MV20IS405)

## ACKNOWLEDGMENT

It gives us immense pleasure to express our sincere gratitude to the management of **Sir M. Visvesvaraya Institute of Technology**, Bengaluru for providing the opportunity and the resources to accomplish our project work in their premises.

On the path of learning, the presence of an experienced guide is indispensable and we would like to thank our guide Vijay Kumar, Professor, Dept. of ISE, for her invaluable help and guidance.

Heartfelt and sincere thanks to Dr.P.Vijay Karthik,HOD, Dept. of ISE, for his suggestions, constant support and encouragement.

We would also like to convey our regards to Dr. V.R. Manjunath, Principal, Sir. MVIT for providing us with the infrastructure and facilities needed to develop our project.

We would also like to thank the staff of the Department of Computer Science and Engineering and lab-in-charges for their cooperation and suggestions. Finally, we would like to thank all our friends for their help and suggestions without which completing this project would not have been possible.

<b>DHARMA PRASAD</b>	<b>1MV20IS400</b>
<b>MANOJ.V</b>	<b>1MV20IS401</b>
<b>SUMANTH . B</b>	<b>1MV20IS403</b>
<b>SUNIL KUMAR G.P</b>	<b>1MV20IS404</b>
<b>VIJAY RAJ SINGH</b>	<b>1MV20IS405</b>

## **ABSTRACT**

Student Management System is software which is helpful for students as well as the school authorities. In the current system all the activities are done manually. Its time saving and scalable. Our Student Management System deals with the various activities related to the students

In the software we can register as a user and user has two types student and administrator. Administrator has the power to add new user and can edit the students details entered. A admin can add students record, attendance status with department wise. All student can search his /her basics details and attendance status with there respective roll number

# CHAPTER-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.

### 1.2 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.

## **CHAPTER-2**

### **STUDY OF EXISTING SYSTEM**

#### **2.1 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.

#### **2.2 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.

## **CHAPTER 3**

### **DATABASE DESIGN**

#### **3.1 SOFTWARE REQUIREMENTS SPECIFICATION**

##### **3.1.1 SOFTWARE REQUIREMENTS:**

- Front end- HTML, CSS, Java Script, Bootstrap
- Back end-Python flask (Python 3.7) , SQLAlchemy,
- Operating System: Windows 10
- Google Chrome/Internet Explorer
- XAMPP (Version-3.7)
- Python main editor (user interface): PyCharm Community
- Workspace editor:Sublime text3

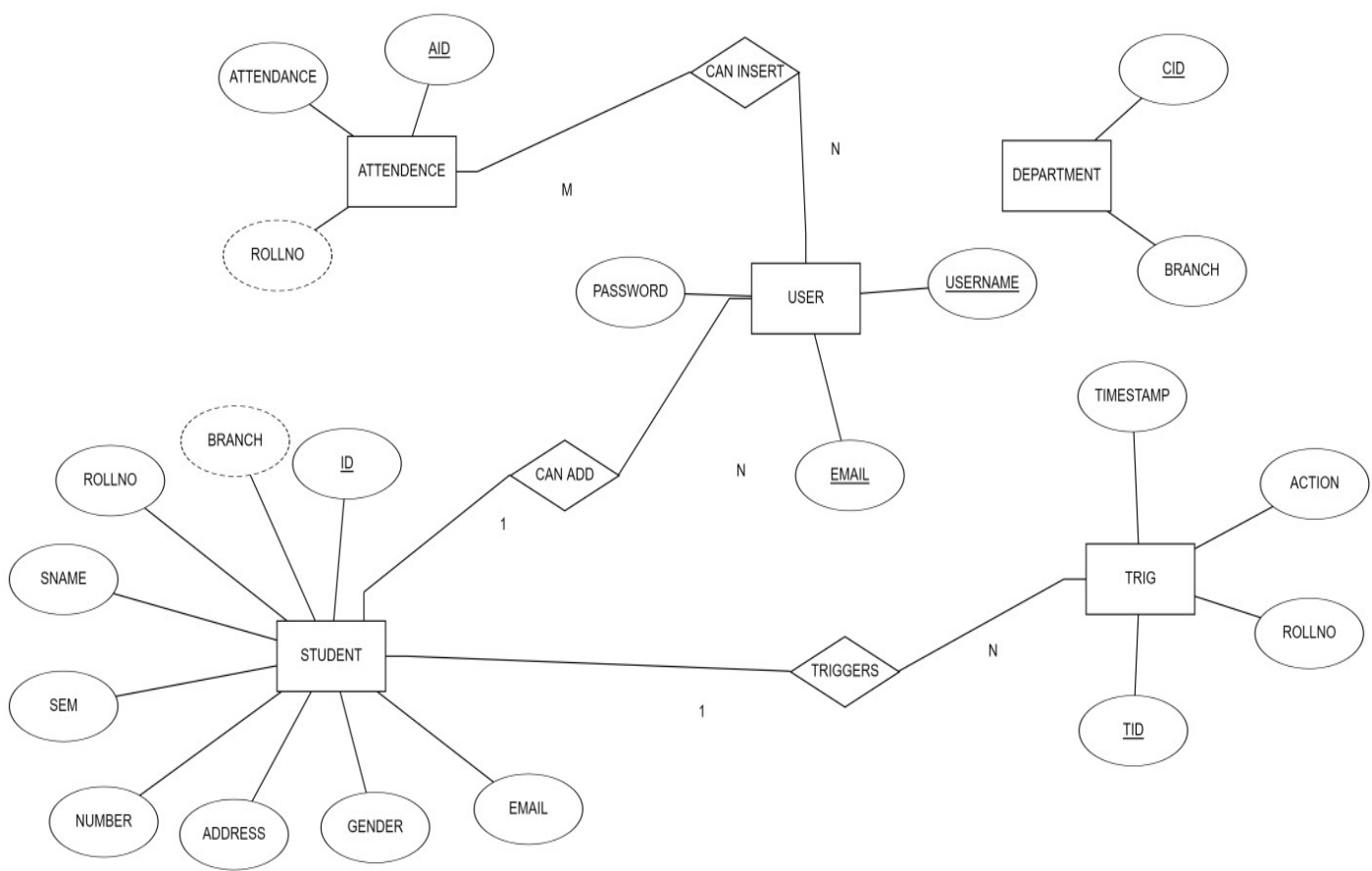
##### **HARDWARE REQUIREMENTS:**

- Computer with a 1.1 GHz or faster processor
- Minimum 2GB of RAM or more
- 2.5 GB of available hard-disk space
- 5400 RPM hard drive
- 1366 × 768 or higher-resolution display
- DVD-ROM drive

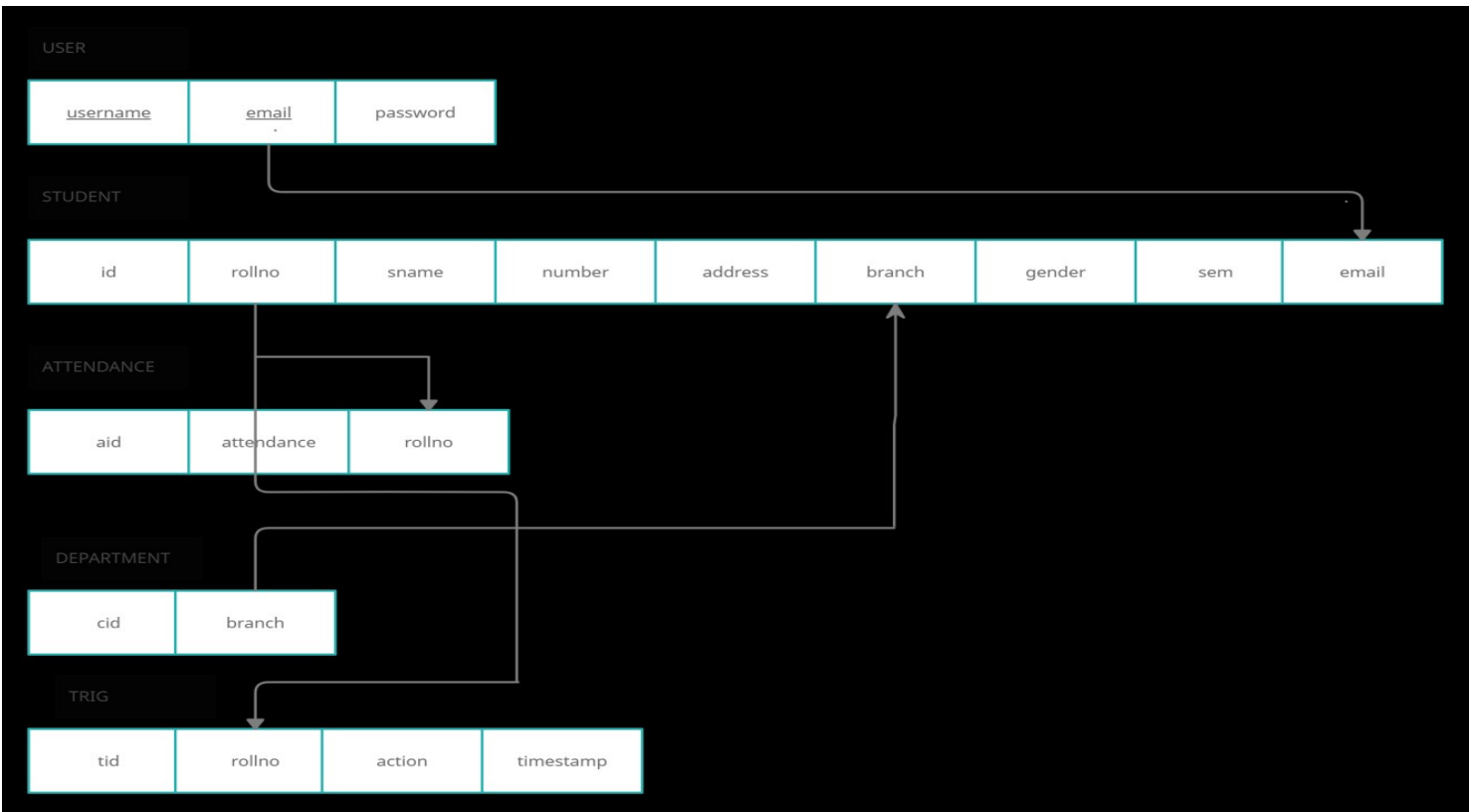


3.2 CONCEPTUAL DESIGN:

3.2.1 E-R DIAGRAM:



3.2.2 SCHEMA DIAGRAM:



## 3.3 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.

### 3.3.1 FRONT END

- HTML(HYPER TEXT MARKUP LANGUAGE)
- CSS
- JAVA SCRIPT
- BOOTSTRAP

### 3.3.2 Back End (MySQL)

- PYTHON FLASK(PYTHON 3.10),SQLAlchemy

### 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 DBMS's include Oracle, DB2, Microsoft Access, Microsoft SQL Server, Firebird, PostgreSQL, MySQL, SQLite, File Maker and Sybase Adaptive Server Enterprise. DBMS's 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, DBMS's 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 DBMS's 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.
- Data security prevents unauthorized users from viewing or updating the database. Using passwords, users are allowed access to the entire database or subsets of it called sub schemas. For example, an employee database can contain all the data about an individual employee, but one group of users may be authorized to view only payroll data, while others are allowed access to only work history and student data.
- If the DBMS provides a way to interactively enter and update the database, as well as interrogate it, this capability allows for managing personal databases. However, it may not leave an audit trail of actions or provide the kinds of controls necessary in a multi-user organization. These controls are only available when a set of application programs are customized for each data entry and updating function.

- ✓ 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. 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)

### 3.3.3 : Triggers

It is the special kind of stored procedure that automatically executes when an event occurs in the database.

Triggers used :

1: Trigger name: on insert

Table: register

Time: after

Event: insert

INSERT INTO trig VALUES(null,NEW.rid,'Farmer Inserted',NOW())

2: Trigger name: on delete

Table: register

Time: after

Event: delete

Definition: INSERT INTO trig VALUES(null,OLD.rid,'FARMER DELETED',NOW())

3: Trigger name: on update

Table: register

Time: after

Event: update

Definition: INSERT INTO trig VALUES(null,NEW.rid,'FARMER UPDATED',NOW())

## 4.2 : Stored Procedure

Routine name: proc

Type: procedure

Definition: Select \* from register;

## 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='hacker'
```

```
# 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 Attendance(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=Attendance(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=Attendance.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}',`em
ail`='{email}',`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">
```

```
<link
href="https://fonts.googleapis.com/css?family=Open+Sans:300,300i,400,400i,700,700i|Raleway:300,
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/style2.css" rel="stylesheet">
{% block style %}
{% endblock style %}

</head>
```

```
<body>
```

```
<!-- ===== Header ===== -->
<header id="header">
    <div class="container">
```



```
<div id="logo" class="pull-left">
```

```
<a href="/" class="scrollto">S.M.S(SIR M.V.I.T)</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="/about">About</a></li>

    <li><a href="/contact">Contact</a></li>
    {% if 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 ===== -->
{% block intro %}
```

```
{% endblock intro %}
```

```
<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 %}
```

```
<!-- 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>
```

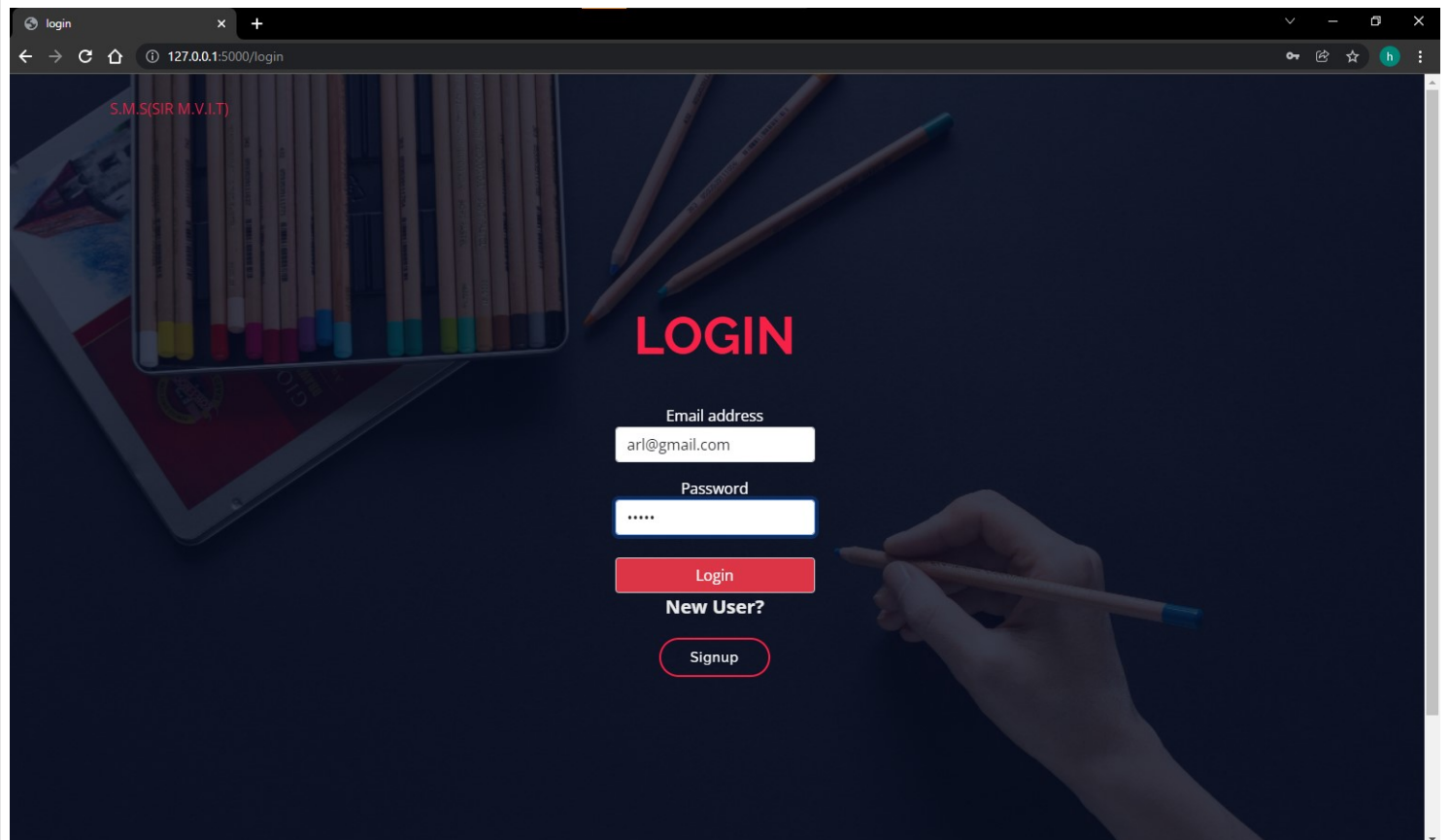
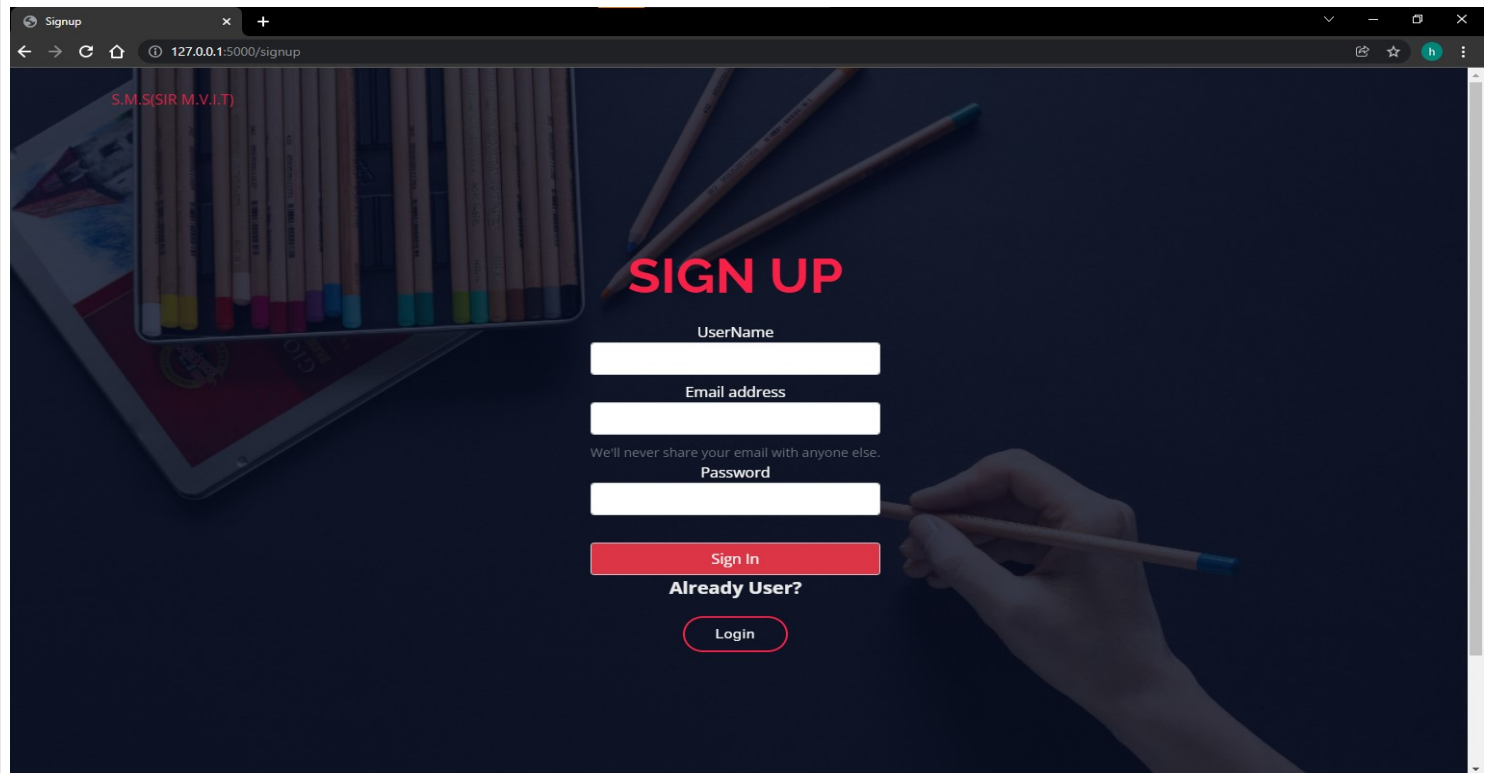
```
</html>
```

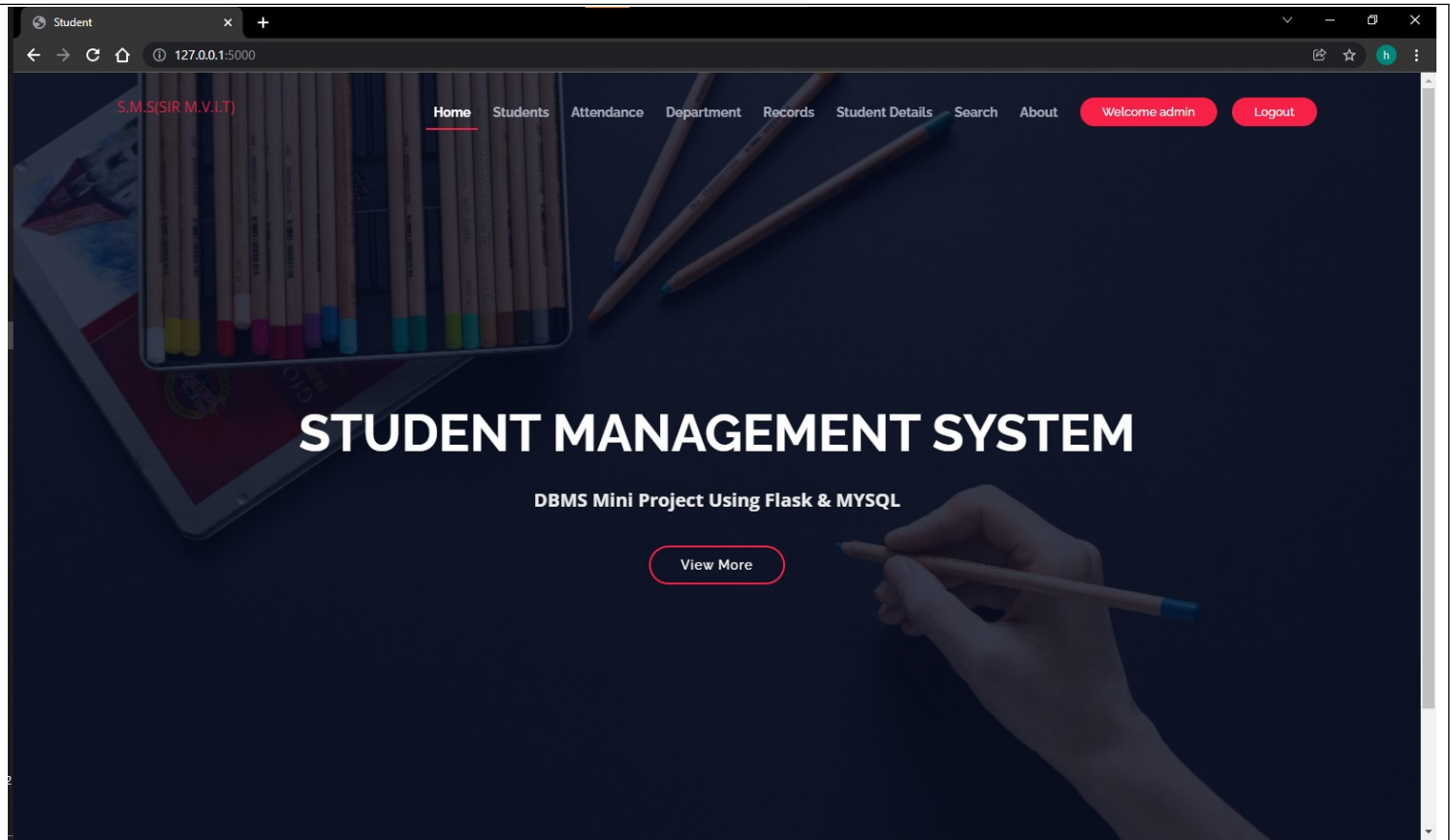
# CHAPTER 4

## USER INTERFACE

### 4.1 SCREEN SHOTS

#### LOGIN AND SIGN UP:





ADD STUDENTS INFO:

Add Students

127.0.0.1:5000/addstudent

S.M.S(SIR M.V.I.T)

HomeStudentsAttendanceDepartmentRecordsStudent DetailsSearchAbout

Welcome adminLogout

Login Success

Roll Number

Student Name

Sem

Select Gender

Select Branch

Email

Phone Number

Address

Student Details

127.0.0.1:5000/studentdetails

S.M.S(SIR M.V.I.T)

Home Students Attendance Department Records Student Details Search About

Welcome admin Logout

# STUDENT MANAGEMENT SYSTEM

DBMS Mini Project Using Flask & MYSQL

View More

## Student Details

SID	ROLL NUMBER	STUDENT NAME	SEM	GENDER	BRANCH	EMAIL	NUMBER	ADDRESS	EDIT	DELETE
8	1	abc	5	male	computer science	abc@gmail.com	1234567892	weihfowieh	Edit	Delete
9	2	bac	5	male	Information Science	bac@gmail.com	1234567893	asfoawfoqwifh	Edit	Delete

TID	ROLL NUMBER	ACTION	TIMESTAMP
10	90	STUDENT INSERTED	2022-01-30 11:31:54
11	1	STUDENT INSERTED	2022-01-30 16:29:30
12	2	STUDENT INSERTED	2022-01-30 16:30:59
13	90	STUDENT DELETED	2022-01-30 16:31:19

ADD ATTENDANCE DETAILS:

Add Attendance

127.0.0.1:5000/addattendance

S.M.S(SIR M.V.I.T)

Home

Students

Attendance

Department

Records

Student Details

Search

About

Welcome admin

Logout

STUDENT MANAGEMENT SYSTEM

DBMS Mini Project Using Flask & MYSQL

View More

Add Attendance Details

1

Attendance Percentage

70

Add Attendance

^

ADD DEPARTMENT:

Add Department

127.0.0.1:5000/department

S.M.S(SIR M.V.I.T)

Home

Students

Attendance

Department

Records

Student Details

Search

About

Welcome admin

Logout

STUDENT MANAGEMENT SYSTEM

DBMS Mini Project Using Flask & MYSQL

View More

Add Department

Enter Department

Add Department

^




Search

127.0.0.1:5000/search

S.M.S(SIR M.V.I.T)

HomeStudentsAttendanceDepartmentRecordsStudent DetailsSearchAbout

Welcome adminLogout



Search Your Details

Enter Your Roll Number

Use Small Letter

Search

Your Details

- Roll No : 1
- Name : abc
- Sem : 5
- Gender : male
- Branch : computer science
- Email : abc@gmail.com
- Number : 1234567892
- Address : weihfowiehlf

Attendance Status

- Attendance : 100

## DATABASE LOCALHOST:

Search

localhost / 127.0.0.1 / studentdbs

localhost/phpmyadmin/index.php?route=/database/structure&server=1&db=studentdbs

phpMyAdmin

Recent Favorites

There are no favorite tables.

New

information\_schema

mysql

performance\_schema

phpmyadmin

studentdbs

- New
- attendance
- department
- student
- test
- trig
- user
- test

Structure

SQL

Search

Query

Export

Import

Operations

Privileges

Routines

Events

Triggers

Designer

Filters

Containing the word:

Table	Action	Rows	Type	Collation	Size	Overhead
<input type="checkbox"/> attendance		2	InnoDB	utf8mb4_general_ci	16.0 KiB	-
<input type="checkbox"/> department		6	InnoDB	utf8mb4_general_ci	16.0 KiB	-
<input type="checkbox"/> student		2	InnoDB	utf8mb4_general_ci	16.0 KiB	-
<input type="checkbox"/> test		1	InnoDB	utf8mb4_general_ci	16.0 KiB	-
<input type="checkbox"/> trig		4	InnoDB	utf8mb4_general_ci	16.0 KiB	-
<input type="checkbox"/> user		1	InnoDB	utf8mb4_general_ci	16.0 KiB	-
6 tables	Sum	16	InnoDB	utf8mb4_general_ci	96.0 KiB	0 B

☐ Check all

With selected:

Print

Data dictionary

Create table

Name:

Number of columns: 4

Go

Console

Search localhost / 127.0.0.1 / studentdb: +

localhost/phpmyadmin/index.php?route=/sql&server=1&db=studentdbms&table=student&pos=0

Server: 127.0.0.1 Database: studentdbms Table: student

Browse Structure SQL Search Insert Export Import Privileges Operations Triggers

Showing rows 0 - 1 (2 total, Query took 0.0004 seconds.)

SELECT \* FROM `student`

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

+ Options

		id	rollno	sname	sem	gender	branch	email	number	address				
<input type="checkbox"/>	Edit	<input type="checkbox"/>	Copy	<input type="checkbox"/>	Delete	8	1	abc	5	male	computer science	abc@gmail.com	1234567892	weihfowieh
<input type="checkbox"/>	Edit	<input type="checkbox"/>	Copy	<input type="checkbox"/>	Delete	9	2	bac	5	male	Information Science	bac@gmail.com	1234567893	asfoawfoqwifh

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

Console

Search localhost / 127.0.0.1 / studentdb: +

localhost/phpmyadmin/index.php?route=/sql&server=1&db=studentdbms&table=user&pos=0

Server: 127.0.0.1 Database: studentdbms Table: user

Browse Structure SQL Search Insert Export Import Privileges Operations Triggers

Showing rows 0 - 0 (1 total, Query took 0.0004 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

+ Options

		id	username	email	password				
<input type="checkbox"/>	Edit	<input type="checkbox"/>	Copy	<input type="checkbox"/>	Delete	5	admin	arl@gmail.com	pbkdf2.sha256:260000\$r6SNAeZUDEOdmubT\$1d23cc2e7734...

Check all With selected: Edit Copy Delete Export

Show all Number of rows: 25 Filter rows: Search this table

Query results operations

Print Copy to clipboard Export Display chart Create view

Console



Search localhost / 127.0.0.1 / studentdb: x +

localhost/phpmyadmin/index.php?route=/sql&server=1&db=studentdbms&table=attendance&pos=0

Server: 127.0.0.1 Database: studentdbms Table: attendance

Browse Structure SQL Search Insert Export Import Privileges Operations Triggers

Showing rows 0 - 1 (2 total. Query took 0.0003 seconds.)

```
SELECT * FROM `attendance`
```

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

+ Options

	aid	rollno	attendance
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	7	90	100
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	8	1	100

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

Console

Search localhost / 127.0.0.1 / studentdb: x +

localhost/phpmyadmin/index.php?route=/sql&server=1&db=studentdbms&table=trig&pos=0

Server: 127.0.0.1 Database: studentdbms Table: trig

Browse Structure SQL Search Insert Export Import Privileges Operations Triggers

Showing rows 0 - 3 (4 total. Query took 0.0004 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

+ Options

	tid	rollno	action	timestamp
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	10	90	STUDENT INSERTED	2022-01-30 11:31:54
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	11	1	STUDENT INSERTED	2022-01-30 16:29:30
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	12	2	STUDENT INSERTED	2022-01-30 16:30:59
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	13	90	STUDENT DELETED	2022-01-30 16:31:19

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

localhost/phpmyadmin/index.php?route=/sql&server=1&db=studentdbms&table=trig&pos=0

## 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 at most 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.

## **FUTURE ENHANCEMENT**

- Enhanced database storage facility
- Enhanced user friendly GUI
- more advanced results systems
- online feedbacks forms