

# **STUDENT MANAGEMENT SYSTEM**

## **A MINI PROJECT REPORT**

*Submitted by*

**Vaibhav Jajodia [RA2011003010128]  
Yashwant Kaushik [RA2011003010135]  
Aniket Mishra [RA2011003010098]**

*Under the guidance of*

**Dr. Sivakumar T K**

Assistant Professor (Sr. Garde), Department of Computer Science and Engineering

*In partial satisfaction of the requirements for the degree of*

**BACHELOR OF TECHNOLOGY**

in

**COMPUTER SCIENCE & ENGINEERING**



**SCHOOL OF COMPUTING**

**COLLEGE OF ENGINEERING AND TECHNOLOGY**

**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**

**KATTANKULATHUR - 603203**

**MAY 2023**



COLLEGE OF ENGINEERING & TECHNOLOGY  
SRM INSTITUTE OF SCIENCE & TECHNOLOGY  
S.R.M. NAGAR, KATTANKULATHUR – 603 203

## **BONAFIDE CERTIFICATE**

Certified that this project report “**STUDENT MANAGEMENT SYSTEM**” is the bonafide work of “**Aniket Mishra [RA2011003010098], Vaibhav Jajodia [RA2011003010128] and Yashwant Kaushik [RA2011003010135]**” of III Year/VI Sem B.tech (CSE) who carried out the mini project work under my supervision for the course 18CSC303J- Database Management systems in SRM Institute of Science and Technology during the academic year 2022-2023(Even sem).

### **SIGNATURE**

Dr. T.K. SIVAKUMAR  
Assistant Professor (Sr. Grade)  
Department of Computing Technologies  
SRM Institute of Science and Technology

## **ABSTRACT**

The Student Management System is a comprehensive software that streamlines academic operations for both students and school administrators. Unlike the current manual system, our software offers automated processes that save time and promote scalability. With our system, users can register as either students or administrators. Administrators have the ability to add and edit student details, including attendance records by department. Additionally, all students can easily access their basic information and attendance status by searching their respective roll numbers. The Student Management System offers a user-friendly platform that optimizes academic operations and facilitates efficient communication between students and administrators.

# TABLE OF CONTENT

Chapter No.	Title	Page No.
	ABSTRACT	3
	TABLE OF CONTENT	4
	LIST OF FIGURES	5
	ABBERRVIATIONS	7
1	INTRODUCTION	8
1.1	Objectives	
1.2	Limitations	
2	STUDY OF EXISTING SYSTEM	9
2.1	Case Study	
2.2	Proposed System	
3	DATABASE DESIGN	10
3.1	Software Requirements	
3.2	Conceptual Design	11
3.2.1	ER Diagram	
3.2.2	Schema Diagram	
3.3	Implementation	12
3.3.1	Back End	15
3.3.2	Front End	21
4	USER INTERFACE	26
	CONCLUSION	33
	FUTURE ENHANCEMENT	34
	REFERENCES	35

## LIST OF FIGURES

<b>Figure No.</b>	<b>Figure Name</b>	<b>Page No.</b>
3.2.1	E-R DIAGRAM	11
3.2.2	SCHEMA DIAGRAM	11
4.1.1 (a)	LOGIN PAGE	26
4.1.1 (b)	LOGIN PAGE	26
4.1.2 (a)	ADD STUDENT INFO	27
4.1.2 (b)	ADD STUDENT INFO	27
4.1.3 (a)	TRIGGERS RECORD	28
4.1.3 (b)	TRIGGERS RECORD	28
4.1.3 (c)	TRIGGERS RECORD	29
4.1.3 (d)	TRIGGERS RECORD	29
4.1.4 (a)	DATABASE LOCALHOST	30
4.1.4 (b)	DATABASE LOCALHOST	30
4.1.4 (c)	DATABASE LOCALHOST	31
4.1.4 (d)	DATABASE LOCALHOST	31
4.1.4 (e)	DATABASE LOCALHOST	32

## **ABBREVIATIONS**

<b>CSS</b>	Cascading Style Sheet
<b>DBMS</b>	Data Base Management System
<b>DDL</b>	Data Definition Language
<b>GUI</b>	Graphics User Interface
<b>HoD</b>	Head of Department
<b>SQL</b>	Structured Query Language
<b>UI</b>	User Interface

# 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 student 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 offers 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 record 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**

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

Frontend- HTML, CSS, Java Script, Bootstrap

Backend-Python flask (Python 3.7), SQL Alchemy,

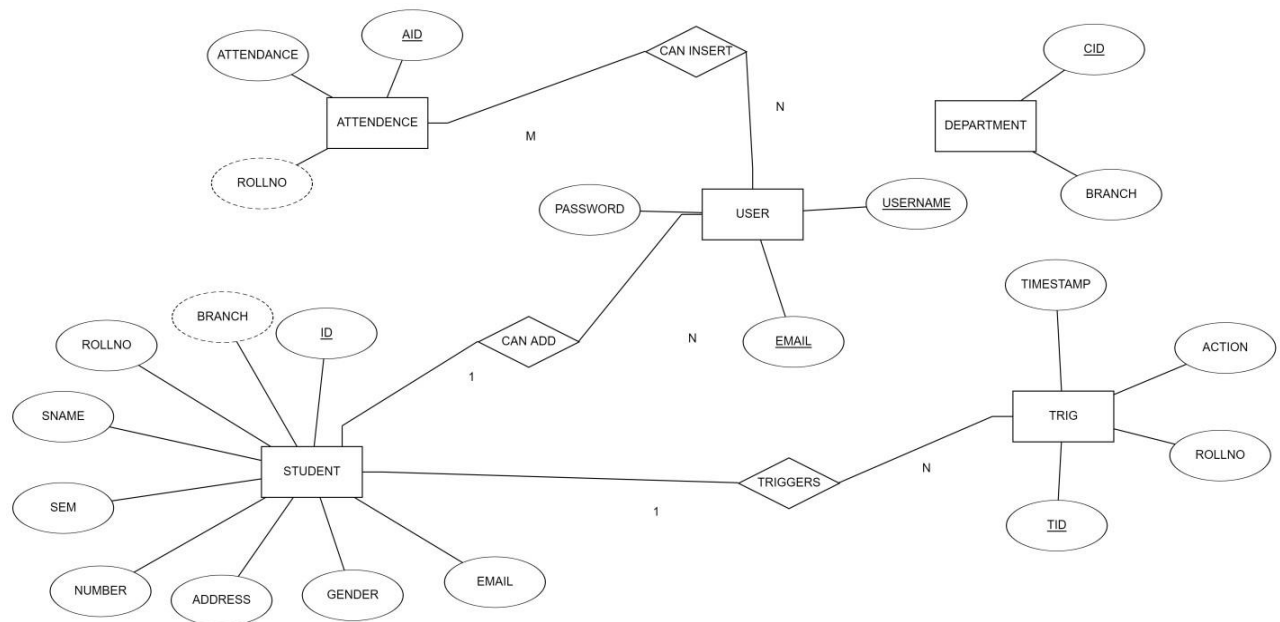
- Operating System: Windows 10
- Google Chrome/Internet Explorer
- XAMPP (Version-3.7)
- Python main editor (user interface): PyCharm Community
- workspace editor: Sublime text 3

##### **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:



The ER (Entity-Relationship) diagram represents the logical structure of the database and illustrates the relationships between entities. In the context of the Student Management System project, the ER diagram would include the following entities and their relationships:

**Student Entity:** This entity represents individual students within the system. It includes attributes such as student ID, name, date of birth, contact information, and address.

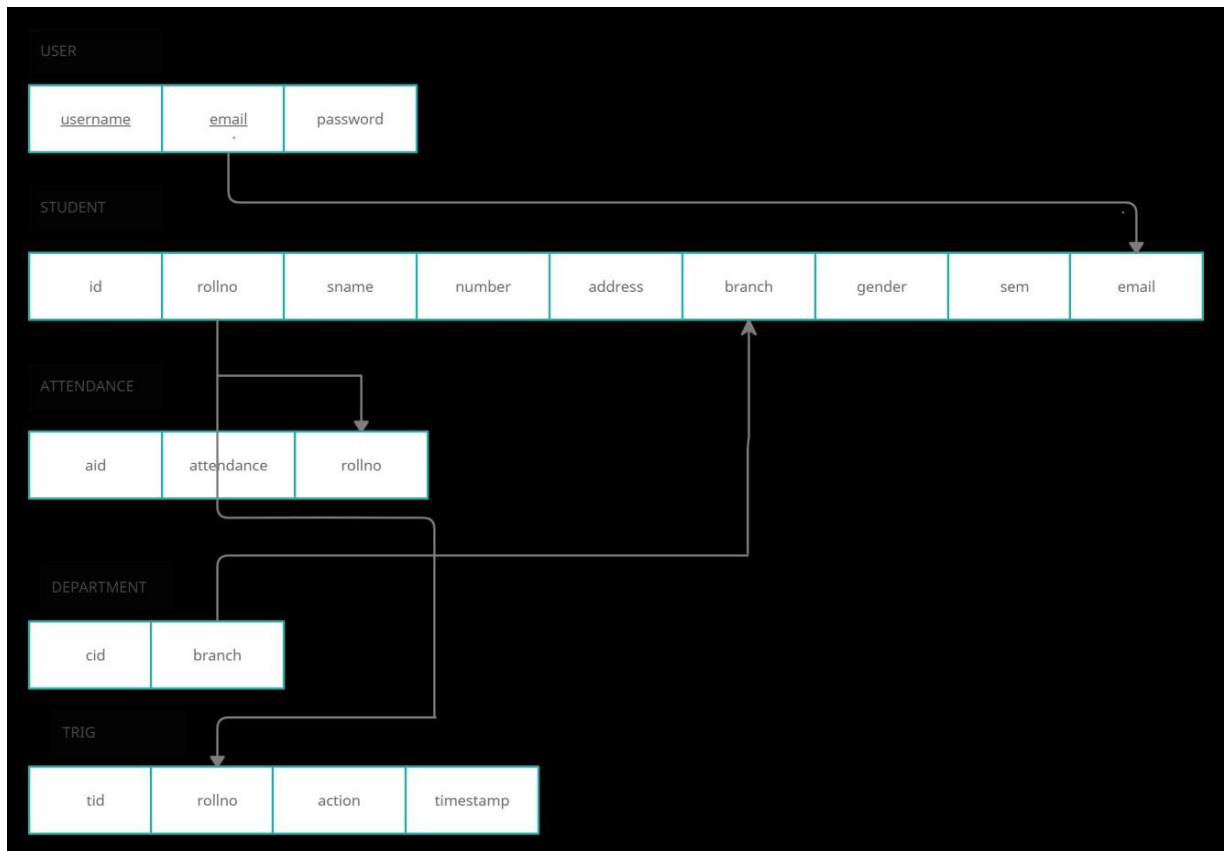
**User Entity:** This entity represents information about the user that is trying to sign in. It includes attributes such as Username, email, Password.

**Department Entity:** This entity represents attributes like Student Id and Branch

**Attendance Entity:** This entity represents the attendance records of students in course sessions. It includes attributes such as attendance ID, student ID, course ID, and session date. It establishes a many-to-one relationship between students and courses since multiple attendance records belong to a specific student and course.

In the ER diagram, these entities are represented as rectangles, and their attributes are listed within each entity. The relationships between entities are indicated by lines connecting the related entities.

### 3.2.2 SCHEMA DIAGRAM:



The schema diagram for the Student Management System project using MySQL and Flask consists of several tables that store different types of information related to students, teachers, courses, and attendance. Let's go through each table and its purpose:

**Students Table:** This table stores information about individual students. It includes attributes such as student ID, name, date of birth, contact information, and address. Each student is uniquely identified by their student ID.

**User Table:** This table stores information about the user that is trying to sign in. It includes attributes such as Username, email, Password.

**Department Table:** The Department table contains attributes like Student Id and Branch

**Attendance Table:** The Attendance table records the attendance of students for each course session. It includes attributes such as attendance ID, student ID, course ID, session date, and a flag to indicate whether the student was present or absent. This table allows tracking of student attendance records for each course session.

The schema diagram depicts the relationships between these tables, which are established through the use of primary and foreign keys.

In summary, the schema diagram for the Student Management System project showcases the structure of the database, illustrating how different entities such as students, attendance are related to each other. This schema provides the foundation for storing and retrieving student-related information efficiently within the system.

### 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.

#### 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.

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.

- 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)

## Stored Procedure

Routine name: proc

Type: procedure

Definition: Select \* from register;

## 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

Definition: INSERT INTO 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())

## 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/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 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}',`email`='{e
m 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 %}

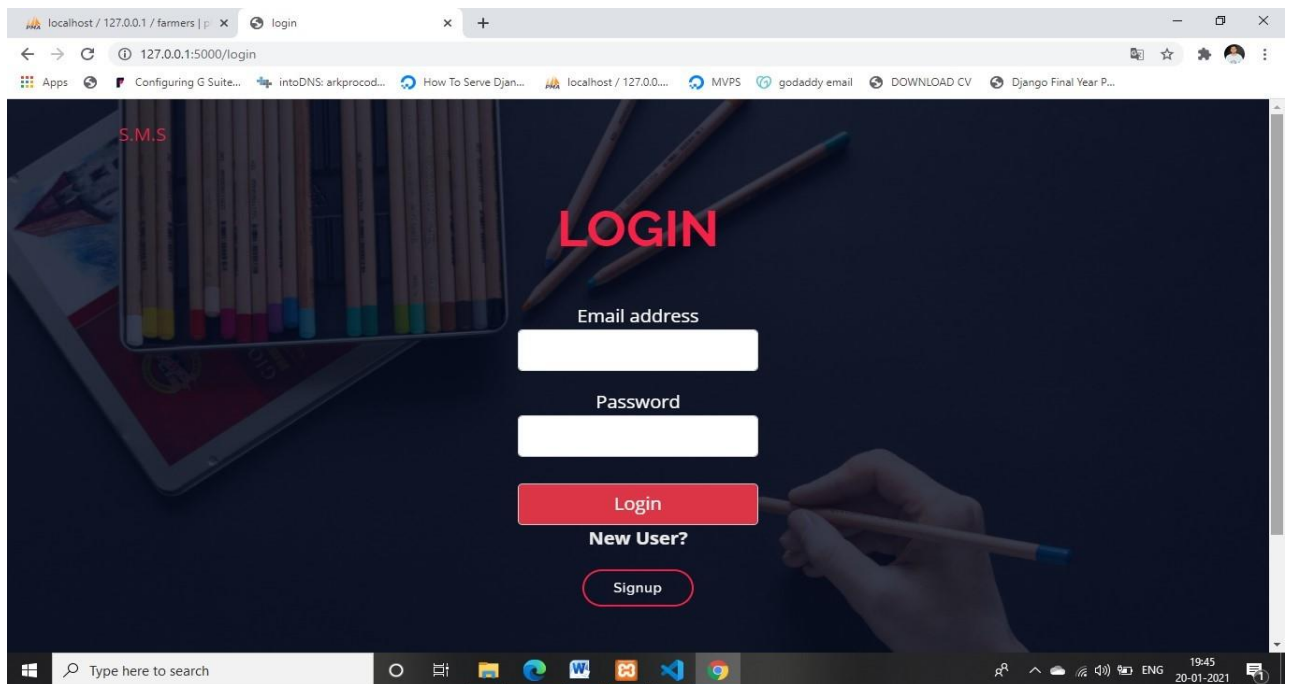
```

## CHAPTER-4

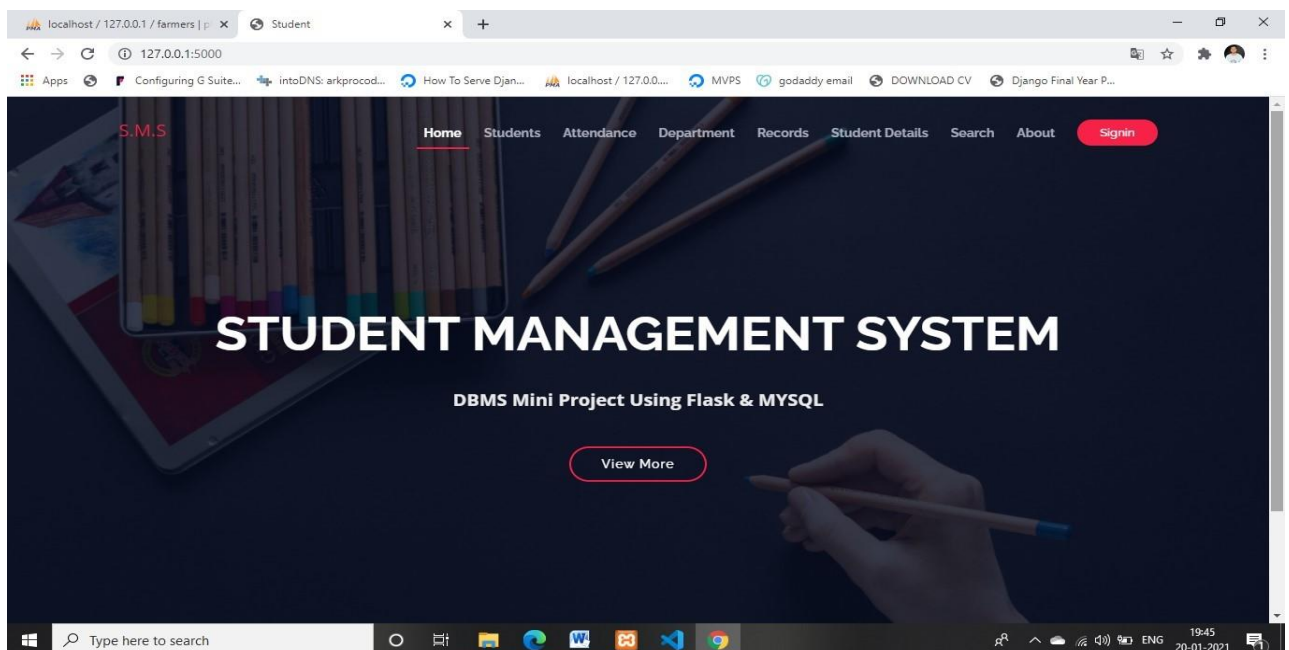
### USER INTERFACE

#### 4.1 SCREEN SHOTS

##### 4.1.1 LOGIN PAGE:



4.1.1 (a)



4.1.1 (b)

### 4.1.2 ADD STUDENT INFO:

The screenshot shows a web browser window with the URL `127.0.0.1:5000/addstudent`. The application has a dark blue header with the logo 'S.M.S' and navigation links: Home, Students, Attendance, Department, Records, Student Details, Search, and About. There are 'Welcome' and 'Logout' buttons on the right. The main content area is titled 'Add Student Details' and contains four input fields: 'Roll Number', 'Student Name', 'Sem', and 'Select Gender'. A red circular button with an upward arrow is located at the bottom right of the form area. The Windows taskbar at the bottom shows the search bar and several application icons.

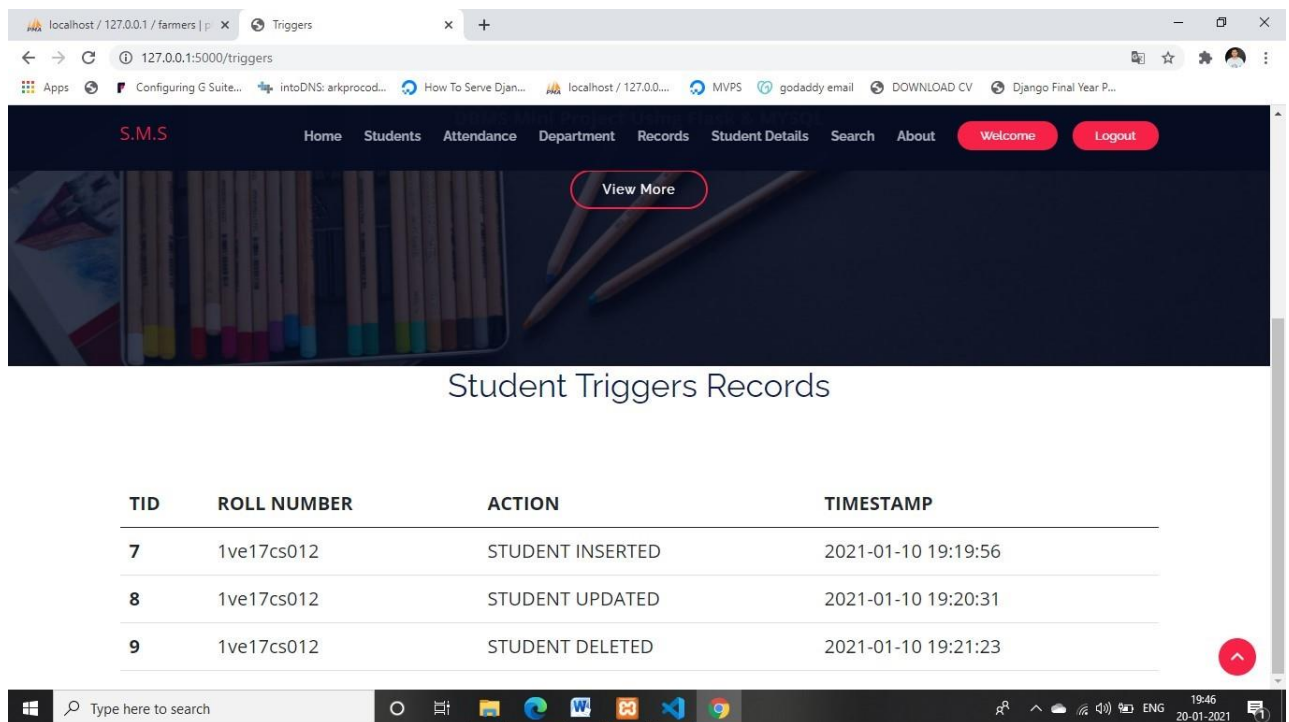
4.1.2 (a)

The screenshot shows a web browser window with the URL `127.0.0.1:5000/studentdetails`. The application header is similar to the previous page, but the 'Welcome' button now says 'Welcome kusuma'. Below the header is a banner for 'DBMS Mini Project Using Flask & MYSQL' with a 'View More' button. The main content area is titled 'Student Details' and displays a table of student information. The table has columns for SID, ROLL NUMBER, STUDENT NAME, SEM, GENDER, BRANCH, EMAIL, NUMBER, ADDRESS, EDIT, and DELETE. There is one row of data for a student named Rohit. The 'EDIT' and 'DELETE' buttons for this row are highlighted with red circles. The Windows taskbar at the bottom shows the search bar and several application icons.

SID	ROLL NUMBER	STUDENT NAME	SEM	GENDER	BRANCH	EMAIL	NUMBER	ADDRESS	EDIT	DELETE
7	1234	rohit	3	male	Electronic and Communication	rohit@gmail.com	9986786453	bangalore	Edit	Delete

4.1.2 (b)

### 4.1.3 TRIGGERS RECORDS:



S.M.S.

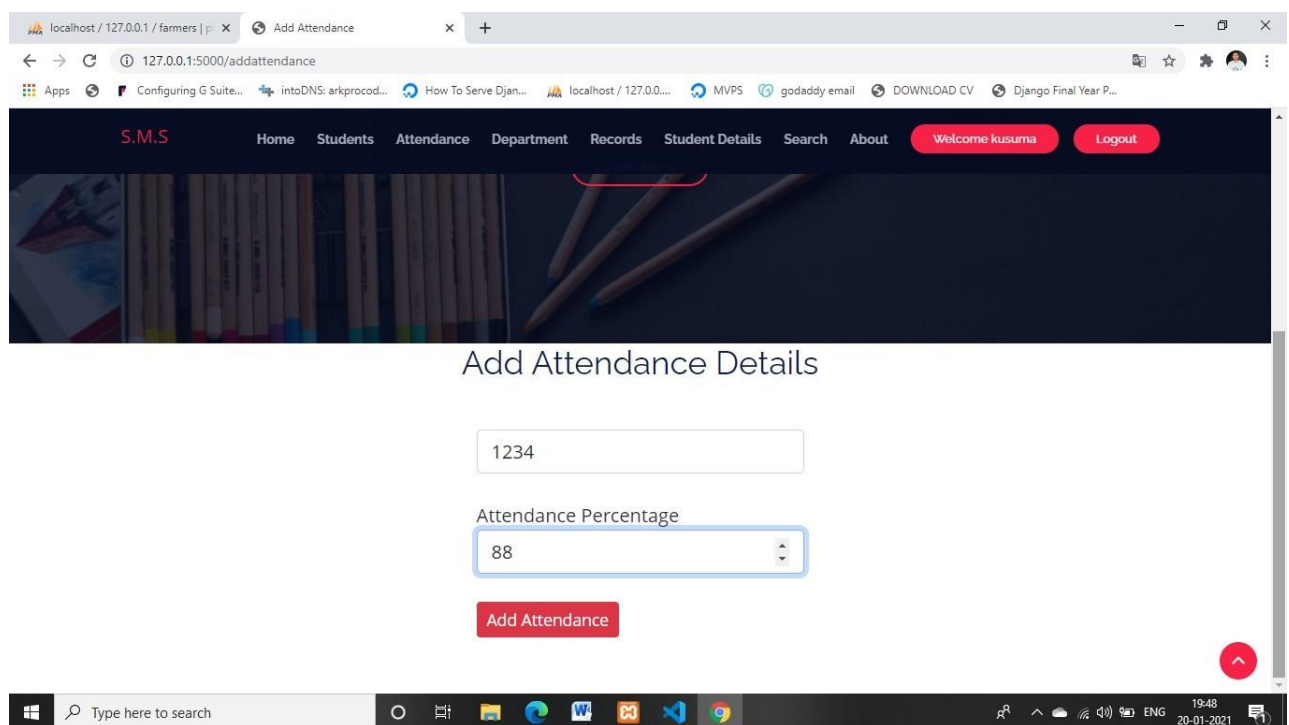
Home Students Attendance Department Records Student Details Search About Welcome Logout

View More

## Student Triggers Records

TID	ROLL NUMBER	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

4.1.3 (a)



S.M.S.

Home Students Attendance Department Records Student Details Search About Welcome kusuma Logout

## Add Attendance Details

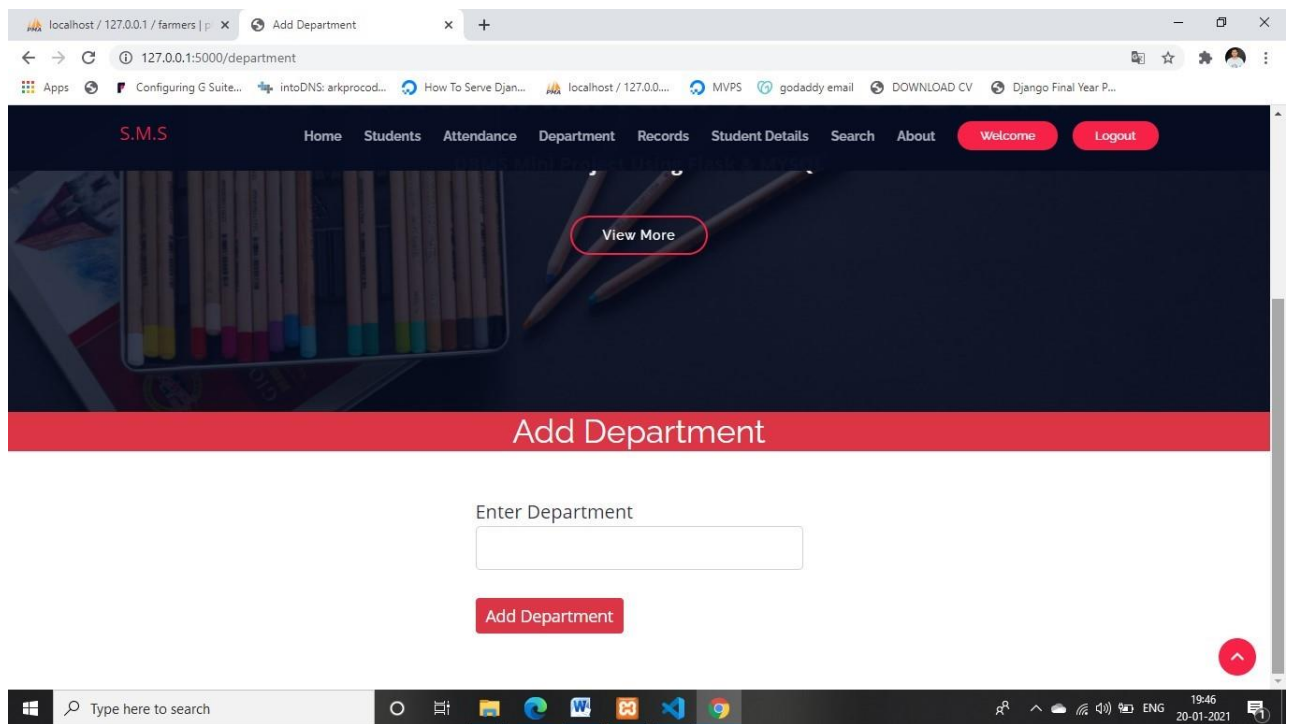
1234

Attendance Percentage

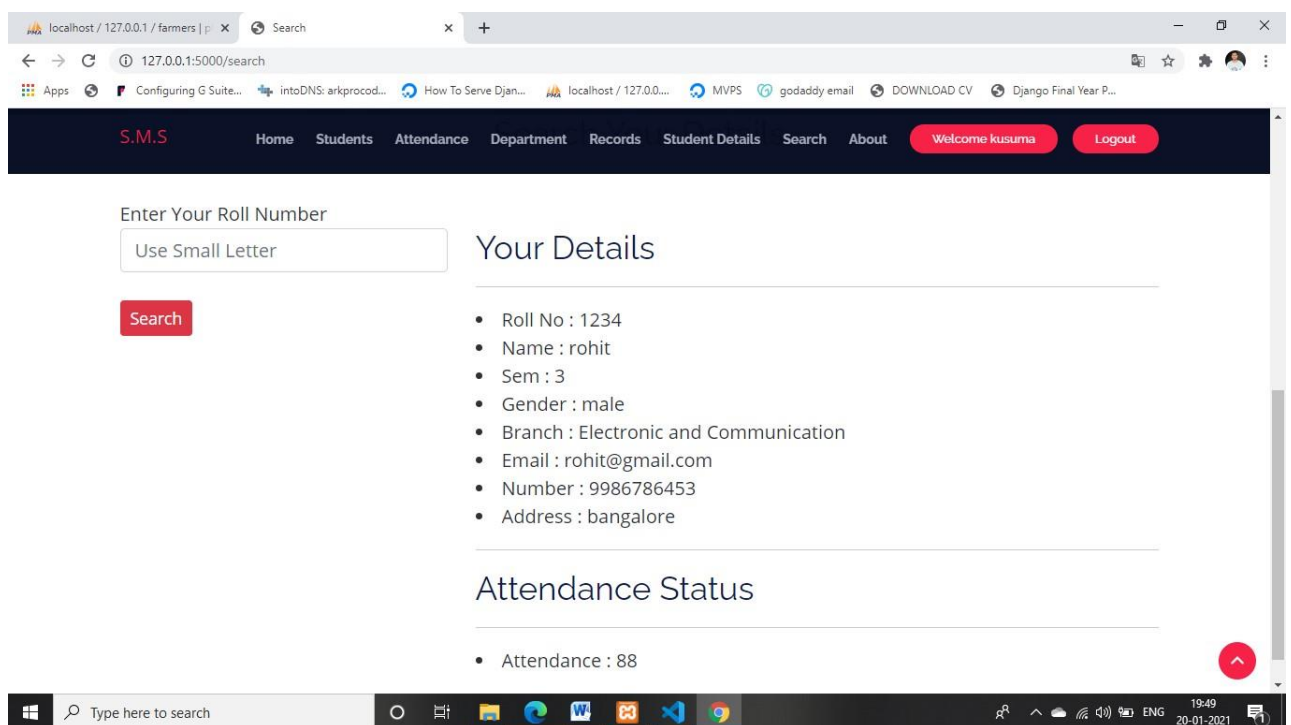
88

Add Attendance

4.1.3 (b)

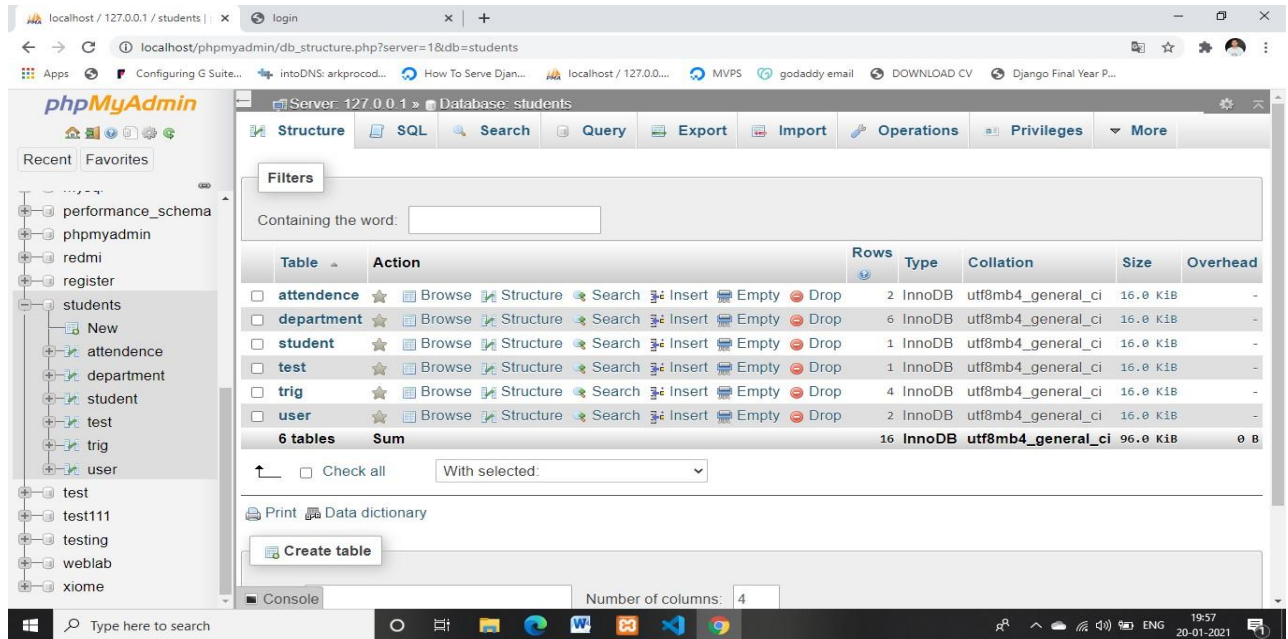


4.1.3 (c)

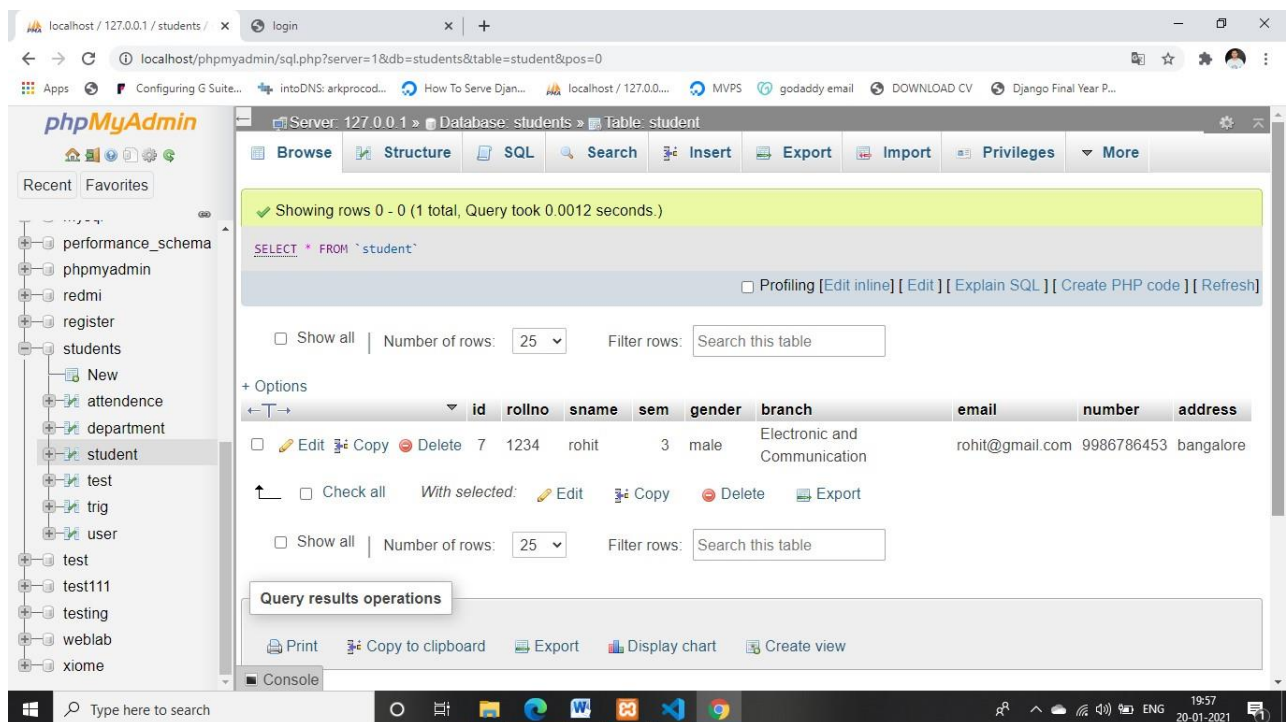


4.1.3 (d)

#### 4.1.4 DATABASE LOCALHOST:

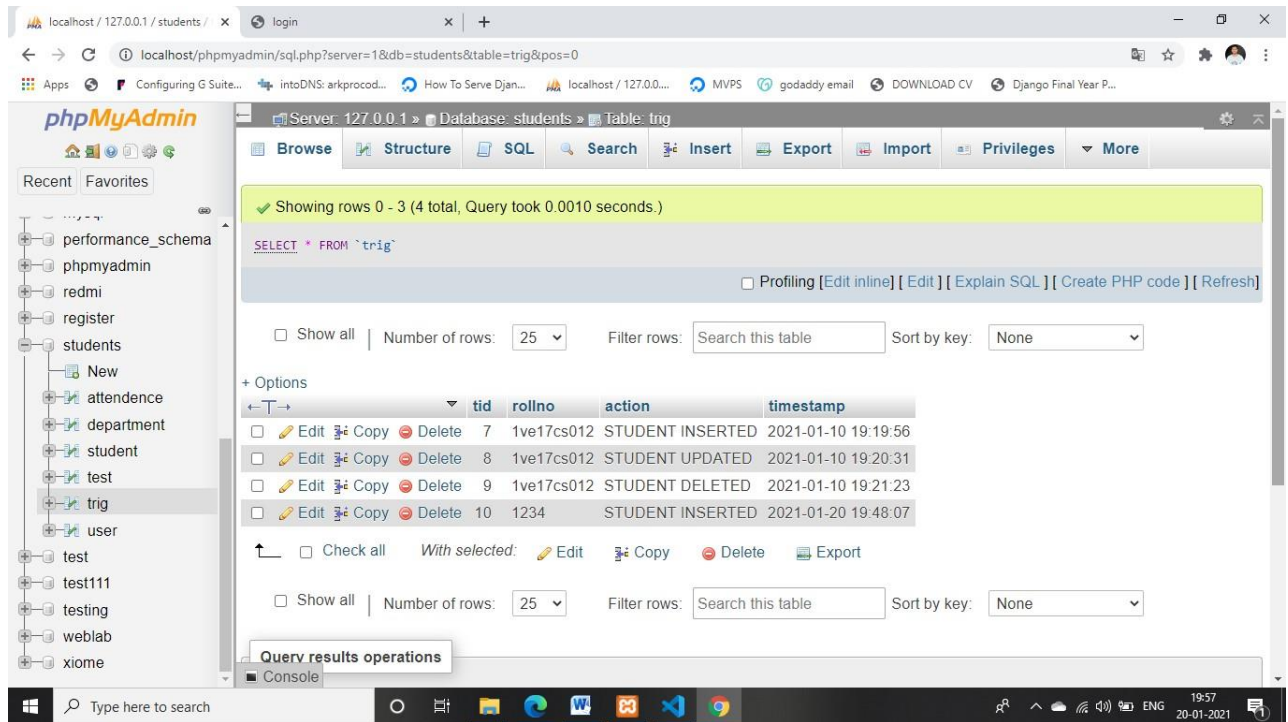


4.1.4 (a)

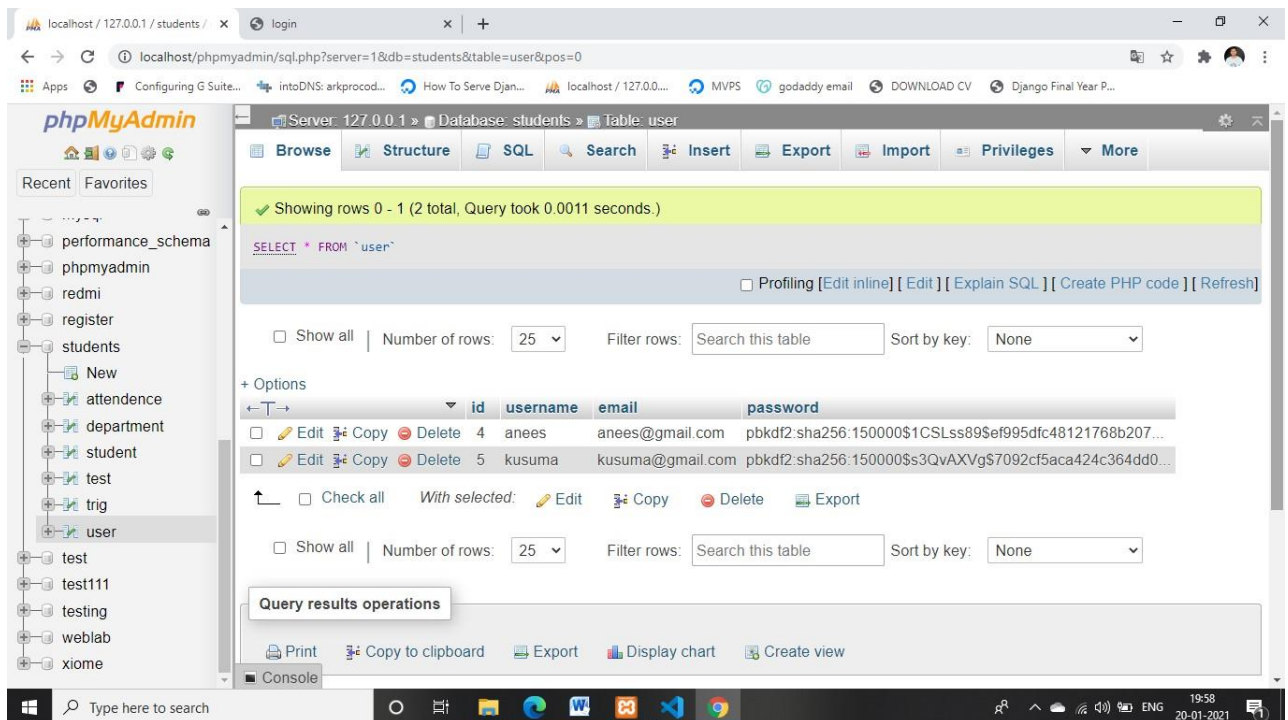


4.1.4 (b)





4.1.4 (c)



4.1.4 (d)

localhost / 127.0.0.1 / students / x login

localhost/phpmyadmin/sql.php?db=students&table=attendance&pos=0

phpMyAdmin

Recent Favorites

performance\_schema  
phpmyadmin  
redmi  
register  
students  
New  
attendance  
department  
student  
test  
trig  
user  
test  
test111  
testing  
weblab  
xiome

Server: 127.0.0.1 » Database: students » Table: attendance

Browse Structure SQL Search Insert Export Import Privileges More

Showing rows 0 - 0 (1 total, Query took 0.0006 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

+ Options

	aid	rollno	attendance
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	7	1234	88

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

Type here to search

19:58 20-01-2021

4.1.4 (e)



# 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

**User Access Control:** Currently, the system does not incorporate a comprehensive user access control mechanism. Implementing role-based access control will allow administrators, teachers, and students to have different levels of access and privileges within the system. This will enhance security and ensure that only authorized individuals can perform specific actions.

**Advanced Reporting:** The project currently provides basic reporting capabilities. Enhancing the reporting module will allow users to generate custom reports based on various parameters such as attendance, grades, and overall performance. Incorporating data visualization techniques such as graphs and charts will provide stakeholders with a visual representation of student progress.

**Mobile Application Development:** Developing a mobile application for the Student Management System will provide on-the-go access to teachers, students, and parents. This application can offer features such as attendance tracking, notifications for important events, assignment submission, and communication between stakeholders. It will enhance the system's accessibility and convenience.

**Student Support System:** Implementing a support system within the Student Management System will enable students to reach out to teachers and administrators for academic assistance or guidance. Integrating a chat or messaging feature will facilitate quick communication and improve the overall learning experience for students.

**Automated Reminders and Notifications:** Enhancing the system with automated reminders and notifications will help students and teachers stay organized and informed. Reminders for upcoming assignments, exams, and events can be sent via email or through push notifications in the mobile application. This will reduce the likelihood of missing important deadlines or events.

## REFERENCES

- <https://flask.palletsprojects.com/>
- <https://dev.mysql.com/doc/>
- <https://flask-mysql.readthedocs.io/>
- <https://docs.sqlalchemy.org/>
- <https://realpython.com/tutorials/flask/>
- "Learning SQL" by Alan Beaulieu
- "Flask Web Development: Developing Web Applications with Python" by Miguel Grinberg