VISVESVARAYA TECHNOLOGICAL UNIVERSITY
"Jnanasangama", Belagavi-590018, Karnataka

BANGALORE INSTITUTE OF TECHNOLOGY
K.R. Road, V.V.Puram, Bengaluru-560 004

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

DATABASE MANAGEMENT SYSTEM MINI PROJECT

18CSL58

STUDENT DETAIL MANAGEMENT PROJECT

Submitted By

ADITYA GAUTAM (1BI19CS009)

AKASH JAIN (1BI19CS011)

for the academic year 2021-22

Department of Computer Science & Engineering

Bangalore Institute of Technology

K.R. Road, V.V.Puram, Bangalore-560 004

VISVESVARAYA TECHNOLOGICAL UNIVERSITY
"Jnanasangama", Belagavi-590018, Karnataka

BANGALORE INSTITUTE OF TECHNOLOGY
K.R. Road, V.V.Puram, Bangalore-560 004

Department of Computer Science & Engineering

Certificate

This is to certify that the implementation of **DBMS MINI PROJECT** entitled "**STUDENT DATABASE MANAGEMENT**" has been successfully completed by

USN: 1BI19CS009 **NAME: ADITYA GAUTAM**

USN: 1BI19CS011 **NAME: AKASH JAIN**

of V semester B.E. for the partial fulfillment of the requirements for the Bachelor's degree in Computer Science & Engineering of the Visvesvaraya Technological University during the academic year 2021-2022.

Lab Incharge
Prof. SUMA L.
Assistant Professor
Dept. of CS&E
Bangalore Institute of Technology
Bangalore

Dr. GIRIJA J.
Professor and Head
Department of CS&E
Bangalore Institute of Technology
Bangalore

ACKNOWLEDGEMENT

The knowledge & satisfaction that accompany the successful completion of any task would be incomplete without mention of people who made it possible, whose guidance and encouragement crowned my effort with success. I would like to thank all and acknowledge the help I have received to carry out this Mini Project.

I would like to convey my thanks to Head of Department **Dr. GIRIJA.** for being kind enough to provide the necessary support to carry out the mini project.

I am most humbled to mention the enthusiastic influence provided by the lab in-charges **Prof. SUMA L.** on the project for their ideas, time to time suggestions for being a constant guide and co-operation shown during the venture and making this project a great success.

I would also take this opportunity to thank my friends and family for their constant support and help. I am very much pleased to express my sincere gratitude to the friendly cooperation shown by all the **staff members** of the Computer Science Department, BIT.

ADITYA GAUTAM (1BI19CS009)

AKASH JAIN (1BI19CS011)

CONTENT

List Of Figures

CHAPTER-1

INTRODUCTION

1.1 Overview

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.

1.2 Objectives

an accurate and flexible system, it will eliminate data redundancy.

- To 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 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.

1.3 Problem Statement

To develop a Student Management System for an organization, especially colleges and schools. 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

CHAPTER 2

BACK-END DESIGN

2.1 Conceptual Database Design

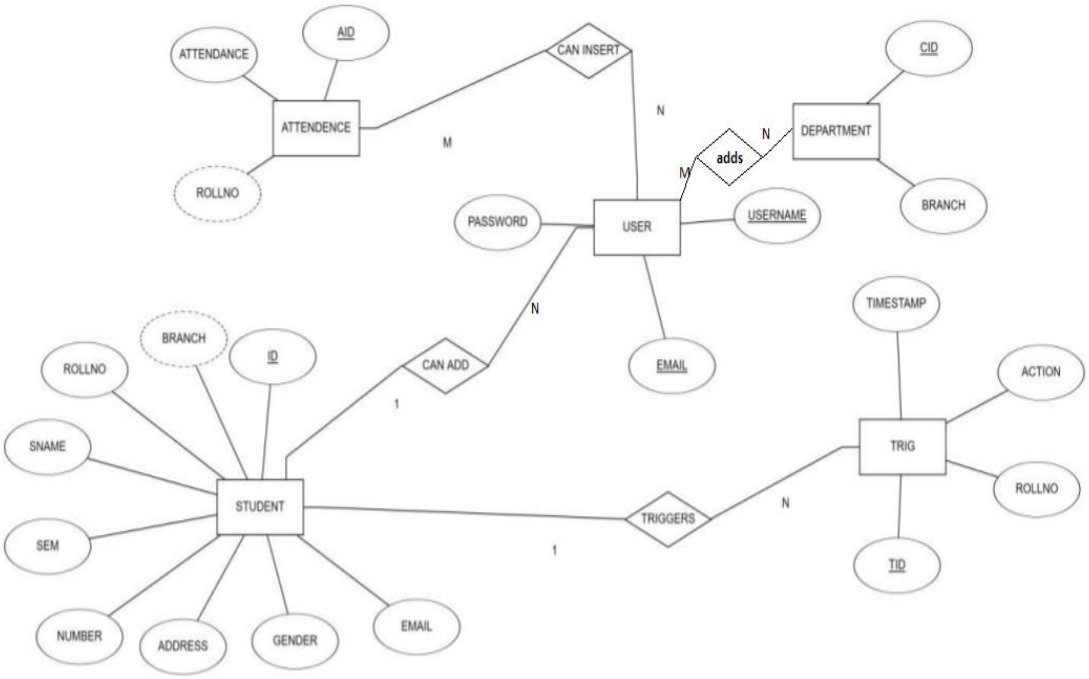

Fig.-2.1.1 E-R Diagram

The ER diagram of the Student Detail Management System shown in fig 2.1.1 has a total of five

entities (User, Department, Attendance, Student, Trig). ATTENDANCE entity is related to USER entity in M to N Relation where attendance can be made for many users. STUDENT and USER has 1 to N relation, means 1 user can add N students. STUDENT entity is related to TRIG entity in 1 to N relation, since there can be multiple times addition, deletion, edition of records. DEPARTMENT entity can be controlled by USER, He can add departments.

2.2 Logical Database Design

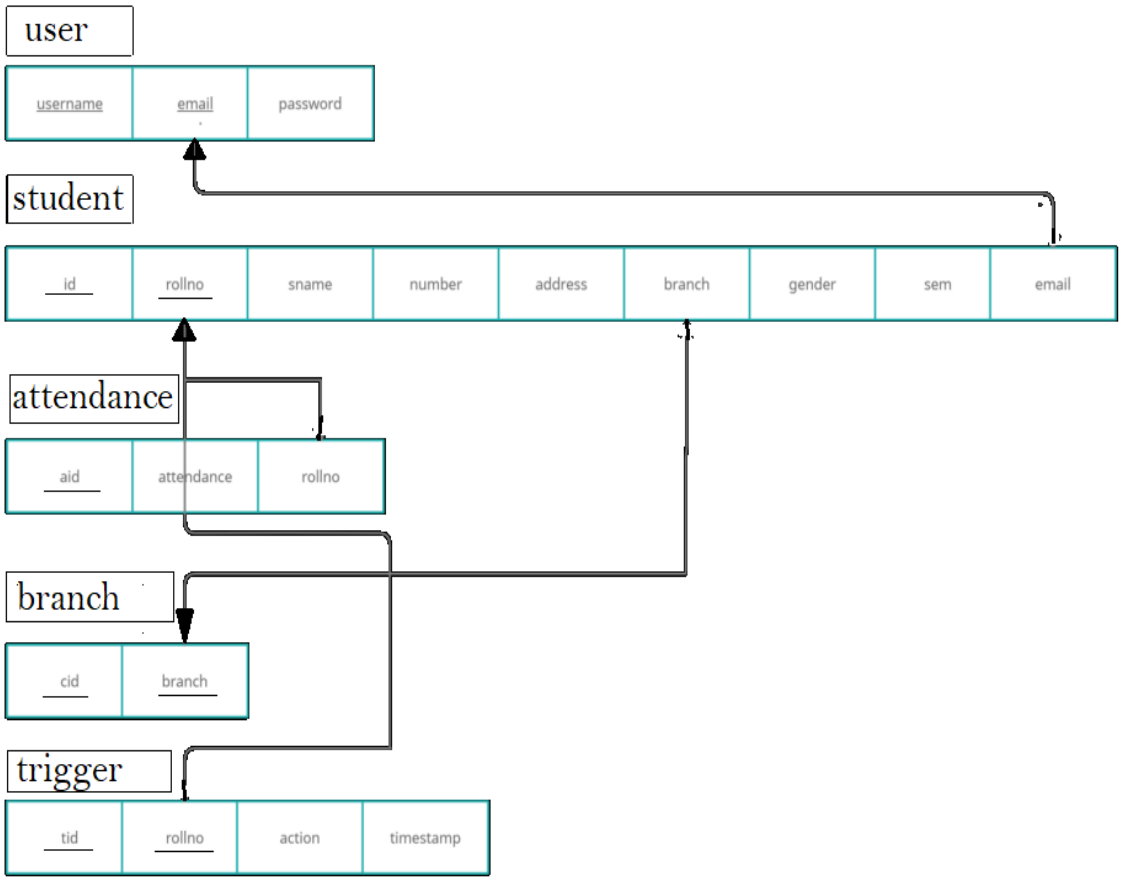

Fig. 2.2.1 E-R Mapping

The database contains a total of five tables, namely - USER, STUDENT, ATTENDANCE, BRANCH, TRIGGER. As shown in fig 2.2.1, the Email attribute in STUDENT has been referenced from USER respectively. The Roll no attribute in ATTENDANCE and TRIGGER has been referenced from STUDENT. Table STUDENT references the attribute Branch from Branch table.

2.3 Normalization

Table 2.3.1 User

Username	Email	Password
Akash123	akash@hotmail.com	Akaash@123

Prime Attribute- Username

Username -> (Username, Email, Password)

1NF- The table satisfies 1NF as all attributes have atomic values.

2NF- The table satisfies 2NF as there are no partial dependencies.

3NF- The table satisfies 3 NF as there are no transitive dependencies.

Table 2.3.2 Student

Roll no	Sname	Phone no	Address	Branch	Gender	Sem	Email
001	Akash	9989001122	vv puram	CSE	Male	5	akash@gmail.com

Prime Attribute – RollNo

RollNo -> (RollNo, Sname, PhoneNo, Address, Branch, Gender, Sem, Email)

1NF- The table satisfies 1NF as all attributes have atomic values.

2NF- The table satisfies 2NF as there are no partial dependencies.

3NF- The table satisfies 3 NF as there are no transitive dependencies.

Table 2.3.3 Attendance

Aid	Attendance	Roll No
5cse001	75	001

Prime Attribute – Aid

Aid -> (Aid, Attendance, RollNo)

1NF- The table satisfies 1NF as all attributes have atomic values.

2NF- The table satisfies 2NF as there are no partial dependencies.

3NF- The table satisfies 3 NF as there are no transitive dependencies.

Table 2.3.4 Branch

Cid	Branch
1bicse	CSE

Prime Attribute – (Cid, Branch)

(Cid, Branch) -> (Cid, Branch)

1NF- The table satisfies 1NF as all attributes have atomic values.

2NF- The table satisfies 2NF as there are no partial dependencies.

3NF- The table satisfies 3 NF as there are no transitive dependencies.

Table 2.3.4 Trigger

tid	Roll no	Action	Timestamp
2	001	Student Updated	09-02-2022 09:12:11

Prime Attribute- (tid, RollNo)

(tid, RollNo) -> (tid, RollNo, Action, Timestamp)

1NF- The table satisfies 1NF as all attributes have atomic values.

2NF- The table satisfies 2NF as there are no partial dependencies.

3NF- The table satisfies 3 NF as there are no transitive dependencies.

2.4 Back End

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:

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)

CHAPTER-3

FRONT-END DESIGN

3.1 Screen Layout

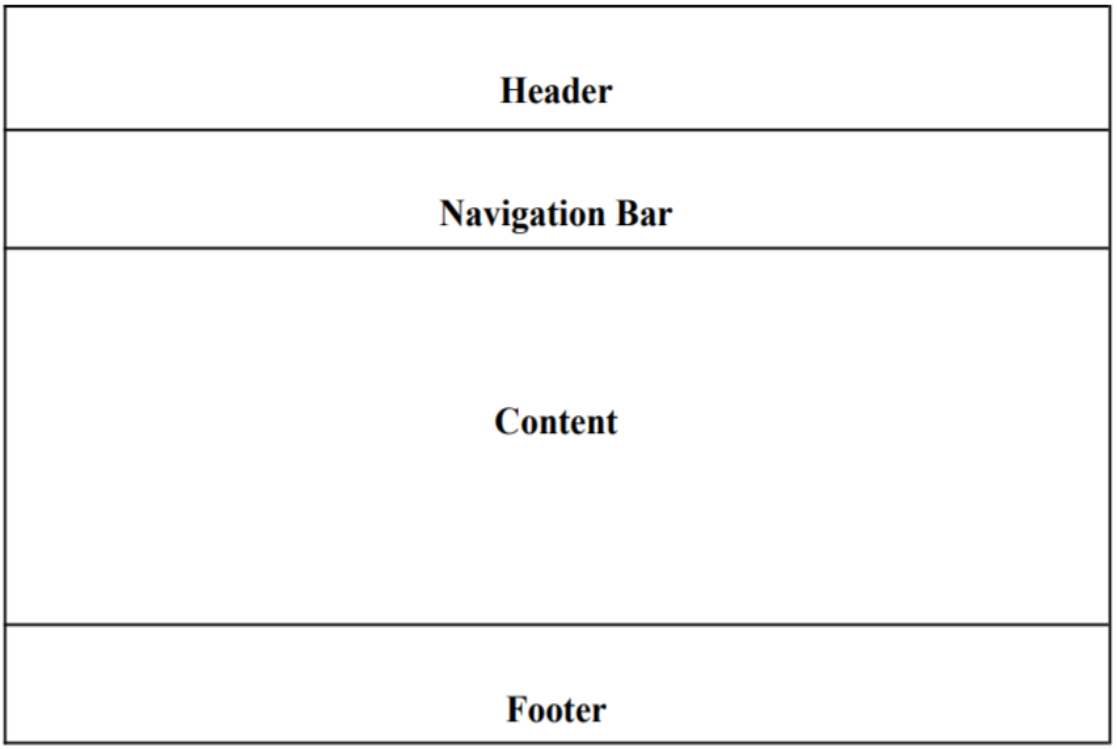

Fig 3.1.1 Screen Layout

One of the most important aspects for any application is the design of the graphical user interface (GUI) and the layout of the screen. Many of the applications that are developed today, because of their good and easy to use interface, have become very popular and are considered to have an attractive design. The basic web page is divided into sections as shown in figure 3.1.1.

- **Navigation-** It contains sign-up form and login form. It contains Home, Students, Attendance, Department, Records, Student Details, Search bars where user can navigate according to his own choice. To see any details, it requires login.

- **Login form-** The login form consists of two fields and one login button. The text fields consist of email where the user enters the email with which he has registered and password where the user enters the password given when he had

registered. The login button posts the data to the backend. It shows an error message ‘Credentials do not match’ whenever either the email or the password or both are wrong. The form also includes a link to the SIGN-UP FORM.

- **Sign-up form-** The user has to register himself first in order to login. The signup form consists of three fields and one sign up button. The fields consist of email, username and password. The signup button pushes the data to the database.

- **Student page-** It contains a form to add student details. It has a form containing fields- Roll No, Sem, Gender, Branch, Phone Number, Email and Address. A user can fill these fields and add student details.

- **Attendance Page**- It contains a select-down bar where we can select the roll no. of students already existing in database and a attendance form where we can add attendance, which will be further updated to Student details database.

- **Search Page-** We can search the details of students using their roll no. If the entered roll no. is not present it will show error message- no such result.

3.2 Front-End

3.2.1 JavaScript

JavaScript often abbreviated JS, is a programming language that is one of the core technologies of the World Wide Web, alongside HTML and CSS. Over 97% of websites use JavaScript on the client side for web page behavior, often incorporating third-party libraries. All major web browsers have a dedicated JavaScript engine to execute the code on users' devices.

JavaScript engines were originally used only in web browsers, but are now core components of some servers and a variety of applications. The most popular runtime system for this usage is Node.js.

3.2.2 HTML

Hypertext Markup Language (HTML) is the main markup language for creating web pages and other information that can be displayed in a web browser. HTML is written in the form of HTML elements consisting of tags enclosed in angle brackets (like<html>), within the web page content. HTML tags most commonly come in pairs like <h1> and </h1, although some tags represent empty elements and so are unpaired, for example <img>. The first tag in a pair is the start tag, and the second tag is the end tag (they are also called opening tags and closing tags). In between these tags web designers can add text, further tags, comments and other types of text-based content.

3.2.3 CSS

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation semantics (the look and formatting) of a document written in a markup language. Its most common application is to style web pages written in HTML and XHTML, but the language can also be applied to any kind of XML document.

CSS is designed primarily to enable the separation of document content (written in HTML or a similar markup language) from document presentation, including elements such as the layout, colors, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple pages to share formatting, and reduce complexity and repetition in the structural content.

CHAPTER-4

MAJOR MODULES

4.1 Login

In the login module there are two labels inside the textbooks namely email-id and password and an arrow button that leads the user to the next event searching page. The user has to enter the email-id and the password to login.

4.2 Student Details

In this module, the user can see which students are register to this application and he can visit to each student details. Their all details can be accessed like semester, branch, attendance, email, roll no.

4.3 Department

In this module user can add the departments, the student can register under a department. HoD, Principal and other authorized staff can access database of students. A user can sign-up and add details of students like semester, branch, attendance, etc.,

CHAPTER-5

IMPLEMENTATION

5.1 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' %}
{% endblock 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 %}
```

Database connection

```
local_server= True
app = Flask( _name)
app.secret_key = ‘adityaakash’
```

5.2Backend Python 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='adityaakash'



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

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


#
app.config['SQLALCHEMY_DATABASE_URL']='mysql://username:password@localhost/datab
as_table_ name'
app.config['SQLALCHEMY_DATABASE_URI']='mysql://root:@localhost/st
udents' db=SQLAlchemy(app)
```

```
# here we will create db models that is
tables class Test(db.Model):
    id=db.Column(db.Integer,primary_key=T
    rue) 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=T
    rue) rollno=db.Column(db.String(100))
    attendance=db.Column(db.Integer())

class Trig(db.Model):
    tid=db.Column(db.Integer,primary_key=Tr
    ue) 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=T
    rue) 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('/studentdetail
 s') def studentdetails():
    query=db.engine.execute(f"SELECT * FROM
    `student`") return
    render_template('studentdetails.html',query=query)

 @app.route('/trigger
 s') def triggers():
    query=db.engine.execute(f"SELECT * FROM
    `trig`") return
    render_template('triggers.html',query=query)

 @app.route('/department',methods=['POST','G
 ET']) def department():
    if request.method=="POST":
        dept=request.form.get('dept')
        query=Department.query.filter_by(branch=dept).f
        irst() 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','G
ET']) 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=atte
        nd) db.session.add(atte)
        db.session.commit()

            flash("Attendance added","warning")
    return render_template('attendance.html',query=query)

@app.route('/search',methods=['POST','G
ET']) 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).f
        irst()
        return

    render_template('search.html',bio=bio,attend=attend)

    return render_template('search.html')

@app.route("/delete/<string:id>",methods=['POST','G
ET']) @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','G
ET']) @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`='{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','G
ET']) 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).fi
        rst() 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=encpassw
        ord) # 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','GE
T']) def login():
    if request.method == "POST":
        email=request.form.get('email')
        password=request.form.get('password')
        user=User.query.filter_by(email=email).fi
        rst()

          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('/logou
t') @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}','{addr
ess}')")



        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)
```

CHAPTER-6

SNAPSHOTS

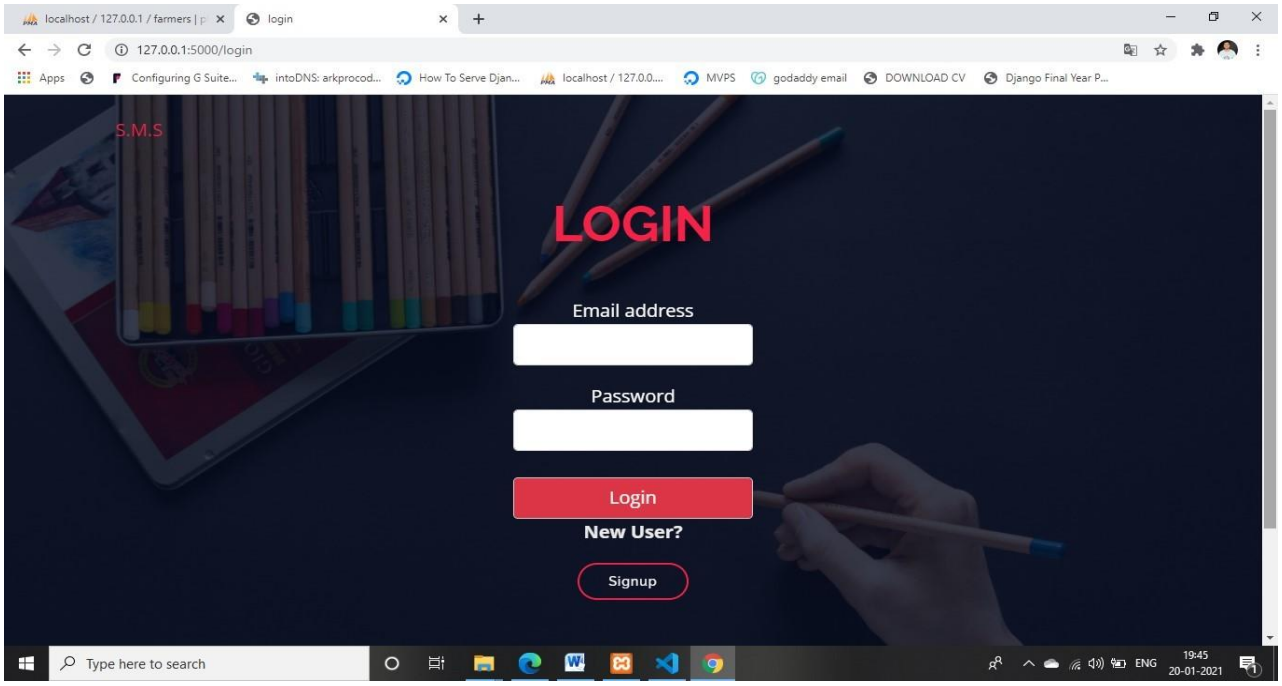

Fig 6.1 Login Page

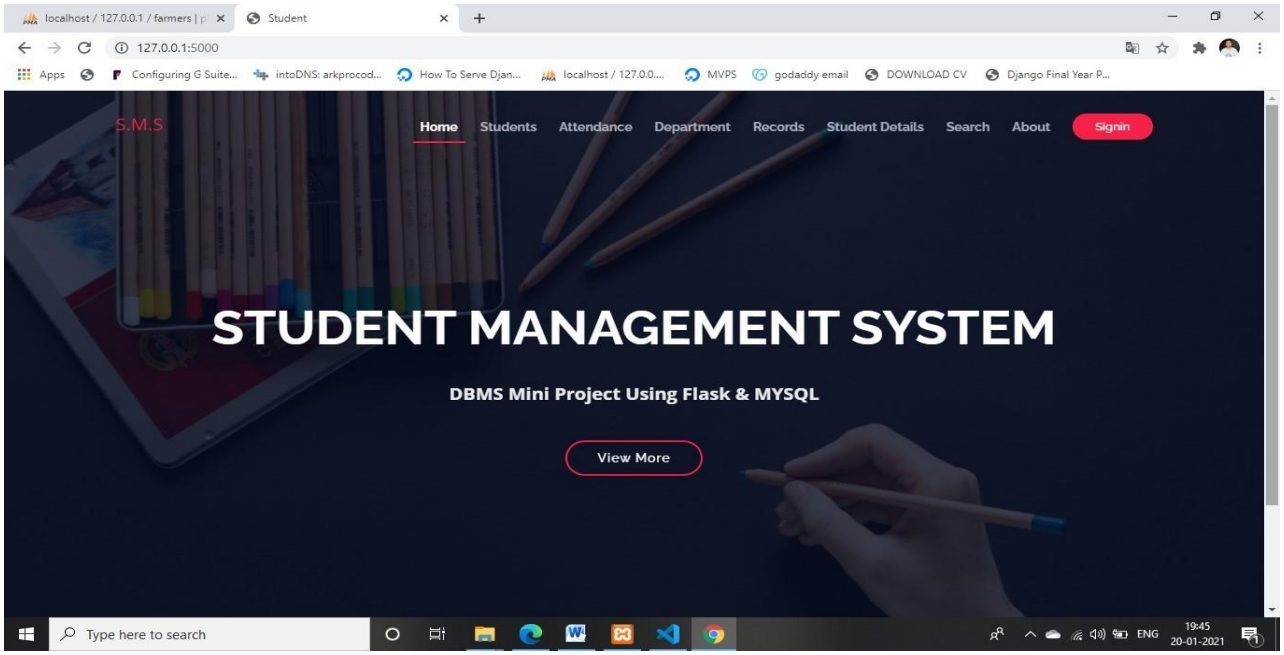

Fig 6.2 Home Page

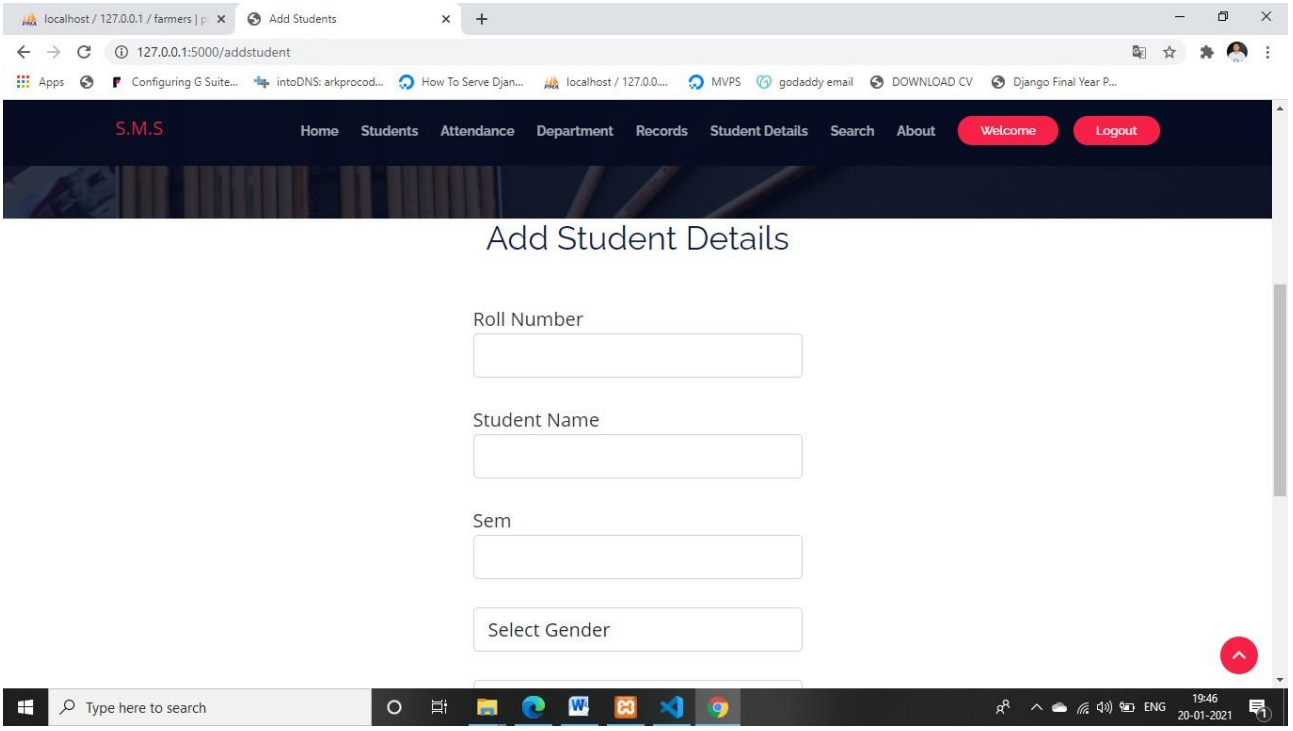

Fig 6.3 Students Page

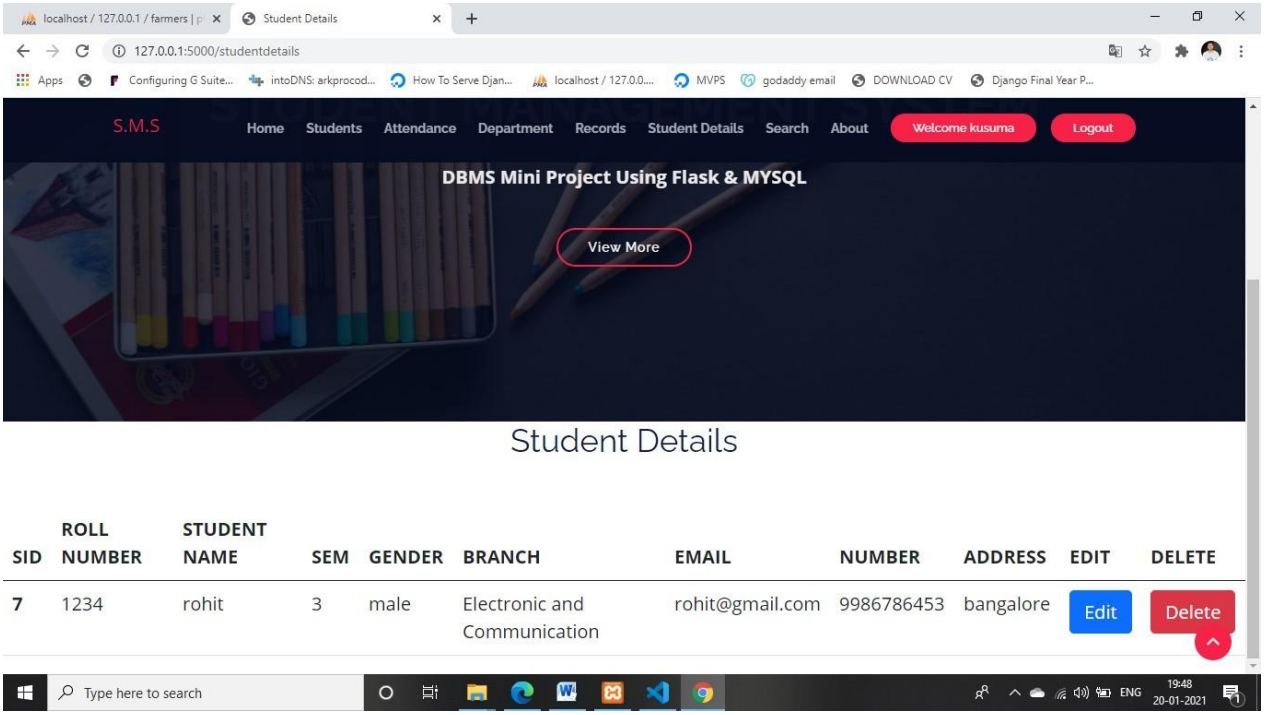

Fig 6.4 Students Detail Page

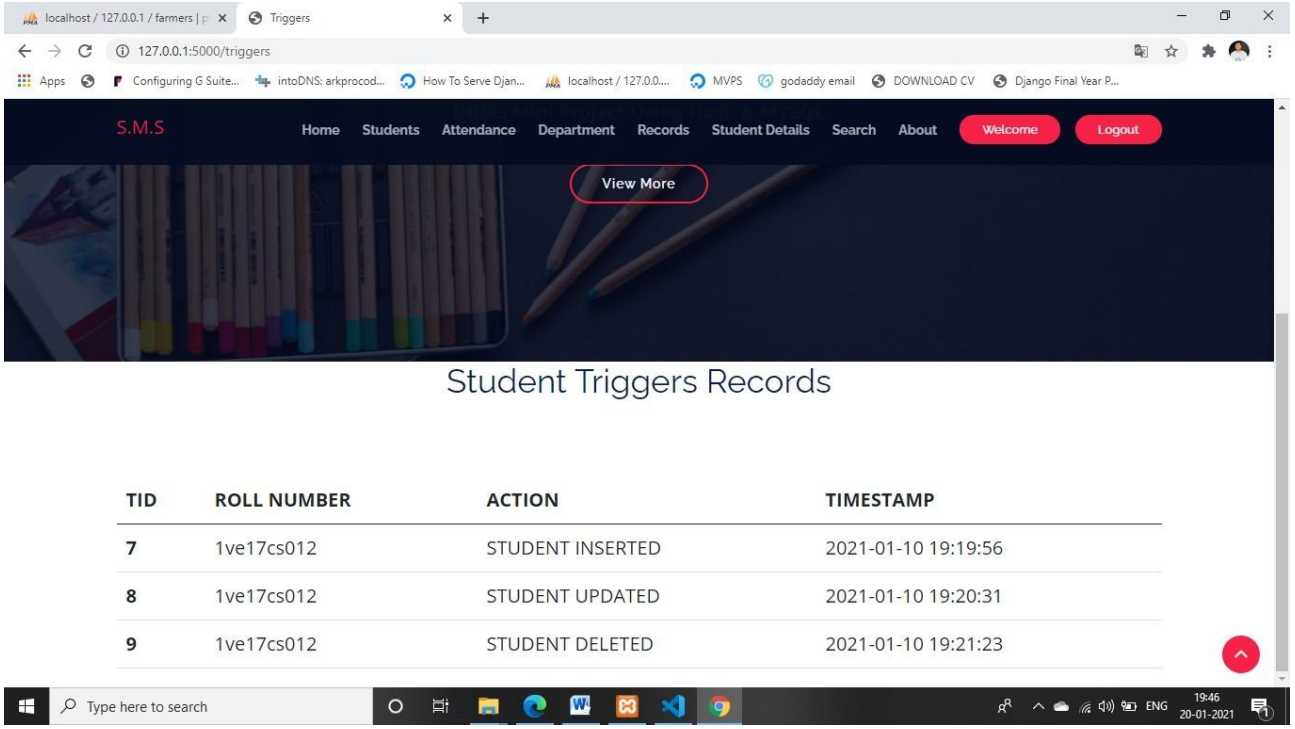

Fig 6.5 Trigger Records

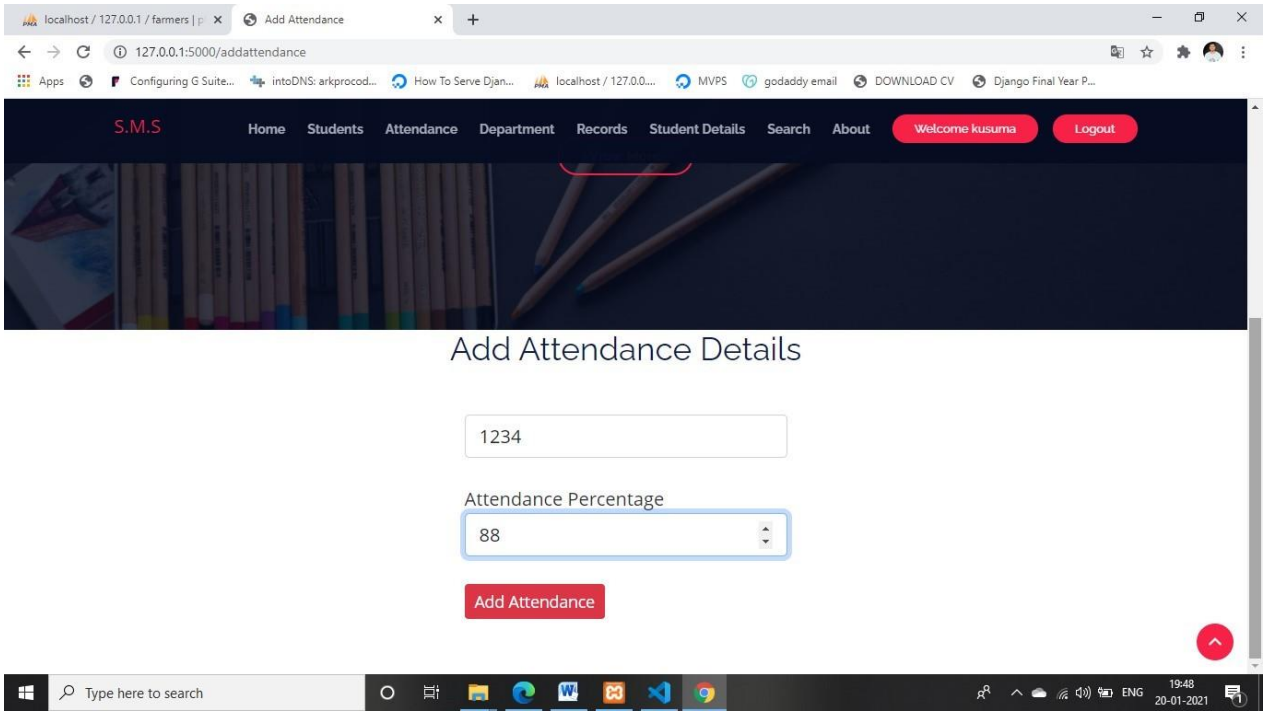

Fig 6.6 Attendance Page

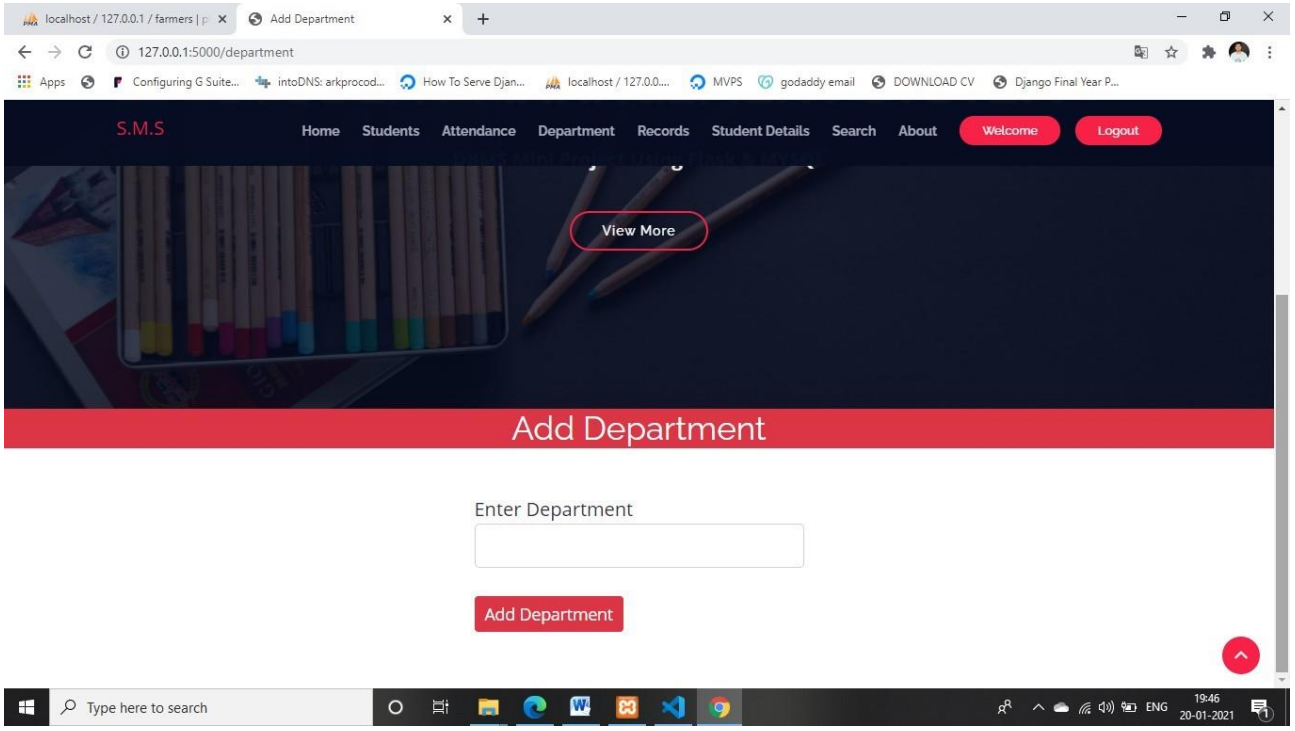

Fig 6.7 Add Department Page

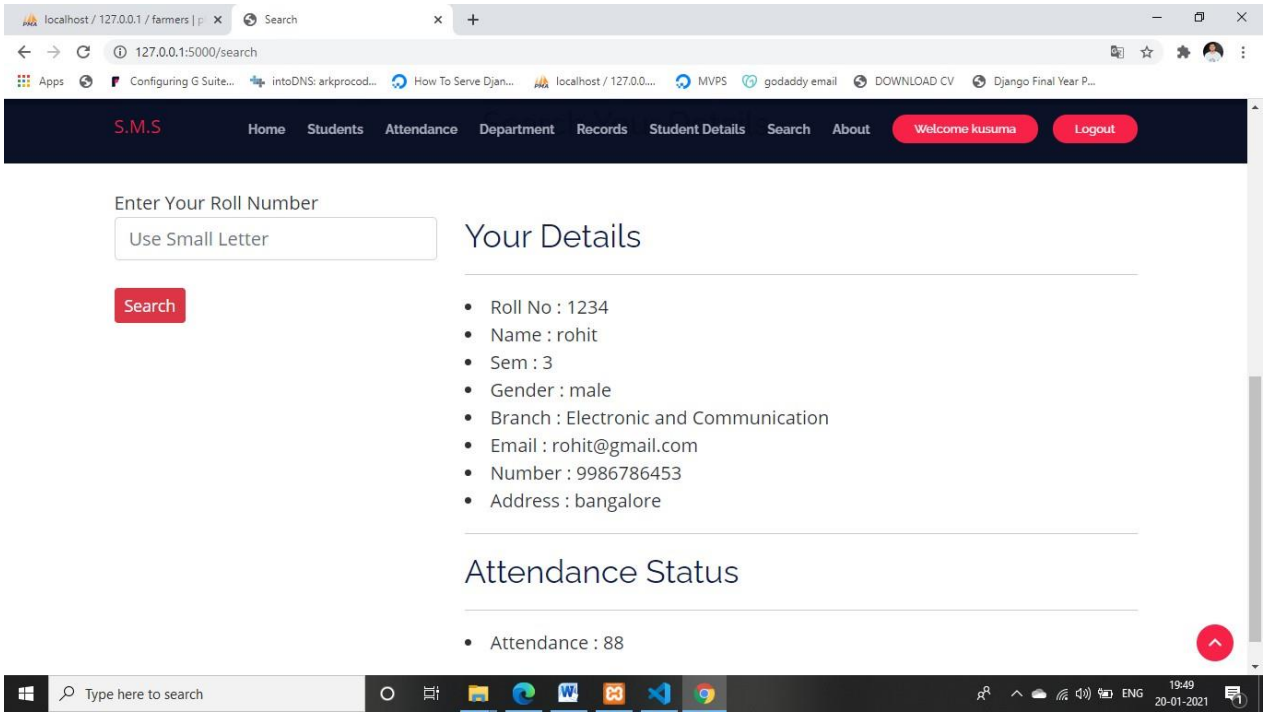

Fig 6.8 Search Page

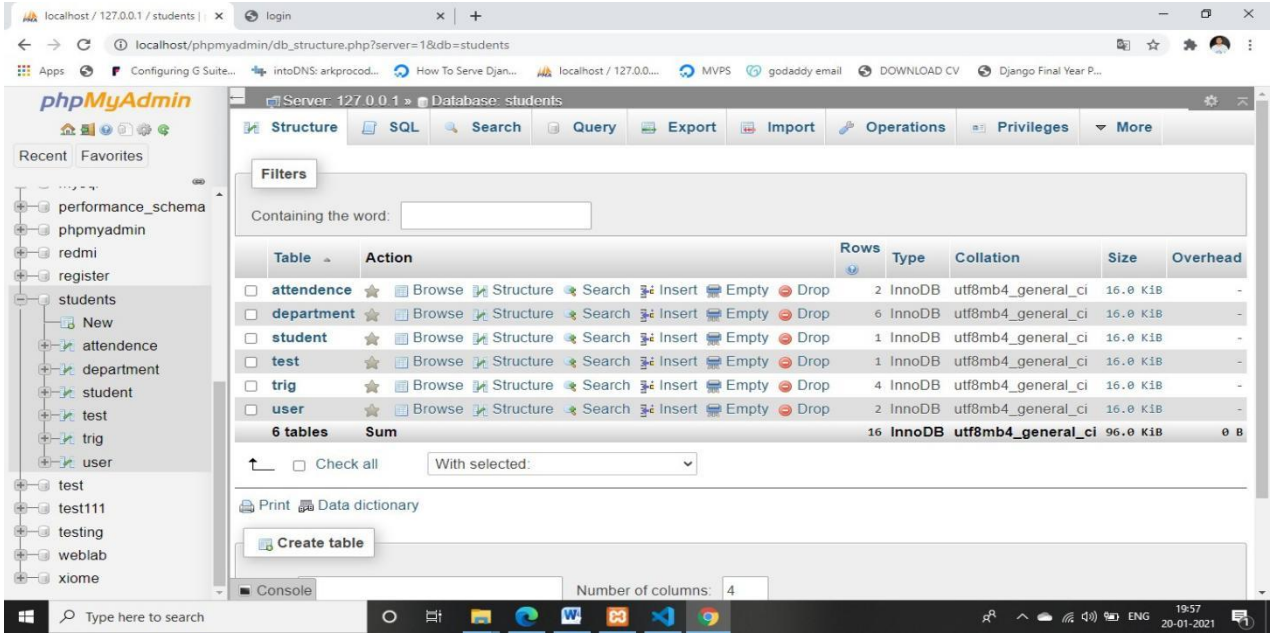

Fig 6.9 Database Tables

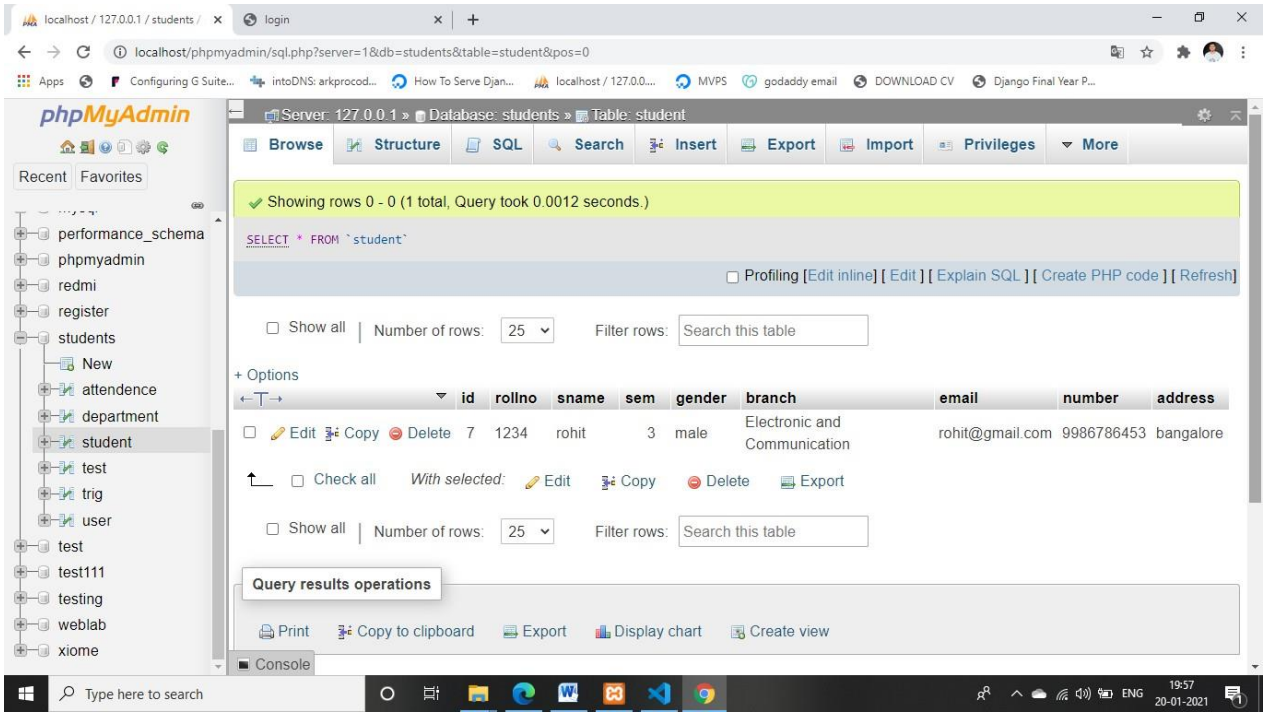

Fig 6.10 Database Rows (Students)

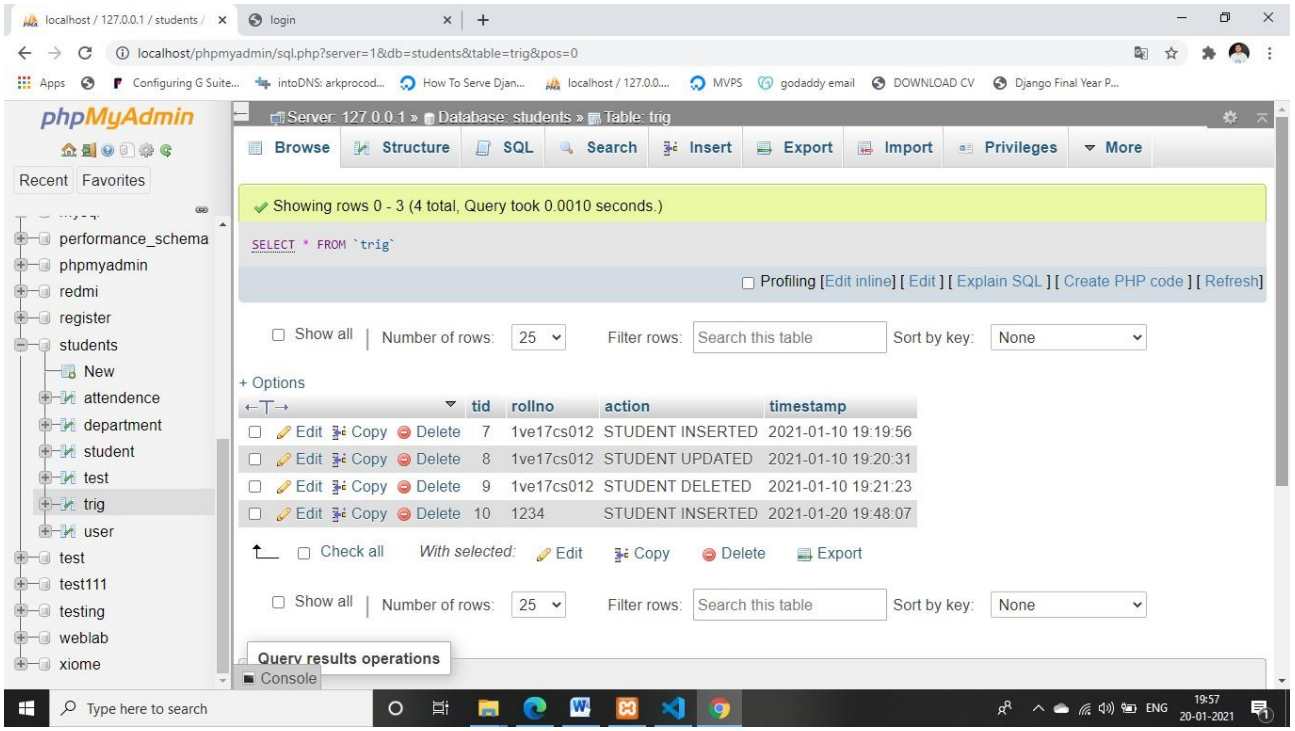

Fig 6.11 Trigger Records (Database)

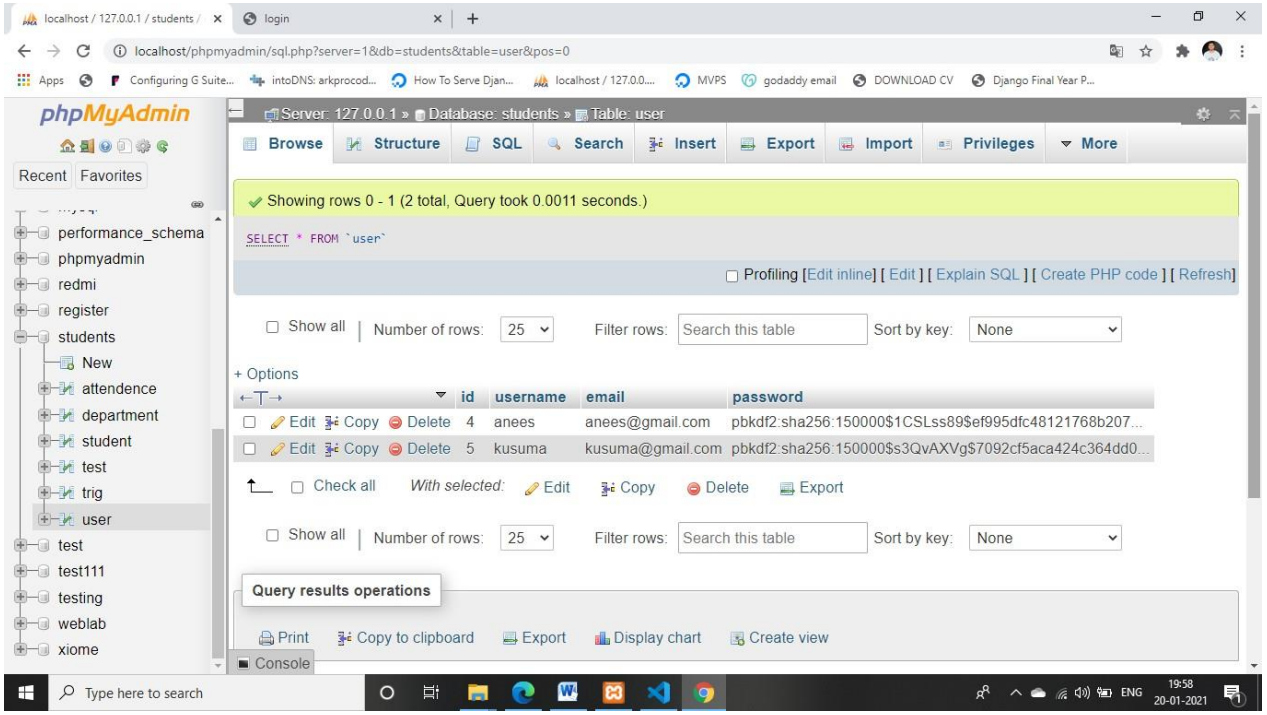

Fig 6.12 User Records

CHAPTER-7

APPLICATIONS

- It can be used by department to add and update student details and attendance respectively.
- Teacher and proctor can fetch and update student attendance and can update it also in respective pages.
- Department can add student details at the time of admission to college.
- Student details can also be deleted whenever student leaves the college and update the Branch, when there is a change of Branch.

CHAPTER-8

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:

Future Enhancement

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