“STUDENT MANAGEMENT SYSTEM USING PYTHON”

A PROJECT REPORT

Submitted by

O.MEENATCHI(953419104033)

P.MALATHI(953419104031)

S.LATHA SELVAM(953419104030)

R.SALMA JESHRAN(953419104044)

In partial fulfillment for the award of the degree

Of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

VV COLLEGE OF ENGINEERING,TISAIYANVILAI

ANNA UNIVERSITY : CHENNAI 600 025



MAY 2022

ANNA UNIVERSITY : CHENNAI 600 025  
BONAFIDE CERTIFICATE

Certified that this project report “STUDENT MANAGEMENT SYSTEM USING PYTHON" is a bonafide work of

**“O.MEENATCHI(953419104033),P.MALATHI(953419104031),S.LATHA SELVAM(953419104030),R.SALMA JESHRAN(953419104044)”** whocarried out mini project work under my supervision.

SIGNATURE SIGNARTURE

Dr. S. Ebenezer Juliet, M.E., Ph.D Ms. D. Merlin Gethsy M.E

HEAD OF THE DEPARTMENT SUPERVISIOR

Computer Science and Engineering Computer Science and Engineering

V V College of Engineering V V College of Engineering

Tisaiyanvilai-627 657 Tisaiyanvilai-627 657

Submitted for the viva-voce held on………………………………………….

INTERNAL EXAMINER EXTERNAL EXAMINER

**ACKNOWLEDGEMENT**

First and foremost, we express our gratitude to God Almighty who has showered blessings on us by giving us the strength and determination to complete this project successfully.

Our sincere thanks to the management for giving us good firmware facilities to complete the project. we thank our department faculty members for their to complete this project.

We sincerely thank our principal **Dr.K.S.Saji M.E.,Ph.D.,** for her valuable guidance and support.

We would like to express our heartfelt gratitude to our Head of the Department **Dr.S.Ebenezer Juliet M.E.,Ph.D.,** for being a constant source of inspiration to us.

We are indebted to our project coordinator **Dr.S.RajaRatana M.E.,Ph.D.,** for providing moral support and motivation.

We thank our project guide **Ms.D.MerlinGethsy AP/CSE .,** for her consistent guidance ,valuable suggestions and constant encouragement that enabled us to execute our project successfully.

We thank our department faculty members for their support to complete this project. Our special thanks to families and friends who have stood by us during the course of our project.

**ABSTRACT**

Our work is to develop a “STUDENT MANAGEMENT SYSTEM” serving both students and teachers. The system deals with efficient and user friendly maintenance of student details. It helps in maintenance of three aspects of a student that is fees, attendance and results. Student can pay the fees and check results and attendance. The teacher can update the attendance of the student based on the presence or absence of in the class which is maintained on the daily basis, check their results, fees status and details, them if failed to maintain the minimum marks. The administrator can check on all three aspects of a student. The user can either be admin , student or teachers .Students can view the details of results in way of increased knowledge.

**KEYWORDS:**

Student Information system, Database, HTML, SQL.

**wh**

**TABLE OF CONTENTS**

**CHAPTER NO TITLE PAGE NO**

1 TABLE OF CONTENTS

* 1. INTRODUCTION 1

1.1.1PURPOSE 1

1.1.2 SCOPE 2

1.2 MODULES 2

2 LITERATURE SURVEY 3

3 SYSTEM ANALYSIS 5

3.1 EXISTING SYSTEM 5

3.2 DRAWBACKS OF EXISTING SYSTEM 5

3.3 PROPOSED SYSTEM 6

3.4 ADVANTAGE OF PROPOSED SYSTEM 6

3.5 APPLICATION SPECIFICATION 6

3.6 MODULE DESCRIPTION 7

3.7 MODULE DESIGN 8

3.8 BLOCK DIAGRAM 9

4 SYSTEM DESIGN 10

4.1 SYSTEM DESIGN 10

4.2 SOFTWARE REQIREMENT 11

4.3 HARDWARE REQUIREMENT 11

4.4 LIBRARIES DEVELOPMENT 11

4.5 PROGRAMMING LANGUAGE 12

4.6 OPERATING SYSTEM SUPPORT 12

5 SYSTEM IMPLEMENTATION 13

5.1 INSTALLATION 13

5.2 SQL SERVER MANAGEMENT STUDIO 13

6 SYSTEM TESTING 16

6.1 SYSTEM TESTING 16

6.1.1 UNIT TESTING 18

6.1.2 INTEGRATION TESTING 18

6.1.3 VALIDATION TESTING 19

6.1.4 OUTPUT TESTING 19

7 RESULT ANALYSIS AND OUTPUTS 20

7.1 LOGIN PAGE 20

7.2 ADD STUDENT INFO 21

7.3 DRIGGERS RECORD 22

7.4 DATABASE LOCAL T 23

8 CONCLUSION AND FUTURRE ENHANCEMENT 24

8.1 CONCLUSION 24

8.2 FUTURE ENHANCEMENT 24

APPENDIX 36

**LIST OF FIGURES**

**FIGURE.NO TITLE PAGE.NO**

3.8 BLOCK DIAGRAM 12

5.1.1 DOWNLOAD VISUAL STUDIO 16

5.1.2 OPEN VISUAL STUDIO 16

5.1.3 START USING VISUAL STUDIO 17

5.2.1 SQL SERVER MANAGEMENT STUDIO 18

5.2.2 MICROSOFT SQL SERVER MANAGEENT STUDIO 19

7.1 LOGIN PAGE 23

7.2 ADD STUDENT INFO 24

7.3 TRIGGERS RECORDS 25

7.4 DATABASE LOCALHOST 26

**LIST OF ABBREVIATIONS**

**SMS** - Student Management system

**LMS -** Learn Management System

**SQL -** Structed Query Language

**HTML-** Hypertext markup Language

**GUI -** Graphical User Interface

**IIS -**Internet Information Server

**IMS -** Instant Messaging System

**IMS -**Information and Monitoring Service

**MMS -**Multimedia Messaging Services

**PHP -** Hypertext processor

**SDLC -**Software Development Life Cycle

**SSI -**Server Side Includes

**SMS -**Short Message Service

**CHAPTER 1**

**INTRODUCTION**

* 1. **INTRODUCTION**

The student Management system is to allow the administrator of any organization to edit and find out the personal details of a students and allows the student to keep up to date his profile. The main purpose of this system is to illustrate the requirement of the project. Student Management System and is intended to help any organization to maintain and manage it’s student profile data. After giving management by every student details are collected by the faculty and manage for each student details.

* + 1. **PURPOSE**

The main purpose of this system is to illustrate the requirements of the project. Student Management System and is intendedto help any organization to maintain the student personal data. The main aim of this system is to reduce the paper work and to eliminate the process and to save staff time. With the help of the system we can gather all the useful information needed to the management in few seconds. This software help the basic information of student automatically. It helps faculty , student and management.

**1.1.2 SCOPE**

This project has been developed will store all the details of the students including their background information, educational qualifications, and all the information needed related to their resume. The student management system is an automated version of manual. The scope of this system is to maintain the all details abouts the student name and faculty portal.

* 1. **MODULES**
* Student Module
* Faculty Module
* Admin Module

**CHAPTER 2**

**LITERATURE SURVEY**

This system basically focuses on providing a simple interface for the easy collation and maintenance of all manner of student information. The criterion and management of accurate up to date information regarding students academic careers is critical students and for the faculties and administration . A student management system deals with all kinds of data from enrollment to graduation, including program of study, attendance record, payment of fees and examination results to name but a few. All these data to need to be made available through an online interface.[1] focusses on providing information to support the operation, management and decision making functions of enterprise . In face huge amount of information , it is required to possess the student information management system to improve the efficiency of student management. In this system a typical student management system will be established to realize the systematization, standardization, and automation of student relationship. [2] focuses on a simple interface for maintenance of student information . The creation and management of accurate, up to date information regarding a students academic career is critically in the educations. Student management system deals with the all kind of student details ,academic reports, college details, course details, curriculum, batch details , placement details, and other resources related details too. It tracks the details of a student which can be queries can be generated based on the vast options related to students, batch, course , completed semester, years also. Different reports and queries can be generated base on their for the entire organization.

**CHAPTER 3**

**SYSTEM ANALYSIS**

**3.1 EXISTING SYSTEM**

Coming to the existing system the details is done by the manual process. In the existing system student database system offer users with a unified view of data from multiple source. To provide a single consistentresult for every object represented in the data sources , data fusion is concerned with revolving data inconsistency. 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 is the major disadvantage of existing system for difficult to handle and the data are not sufficient.

**3.2 DRAWBACKS OF EXISTING SYSTEM**

* Slower processing.
* Greater risk of errors.
* Difficulty in making corrections and alterations.
* Less consistent quality of output.
* Maintain the details was very difficult.
* Long time maintain the records was tough one .

**3.3 PROPOSED SYSTEM**

In the proposed system of they have proposed that researchers adopt the below definition to allow for the coherent development of theory in the management. In order to have successful student management system we need to make many decisions related to the flow of marks, attendance , and some data. Each records should be added in a way to increase the scalability, student management is more complex in colleges and other universities because the impact of people number requiring adequate information of student need.

**3.4 ADVANTAGE OF PROPOSED SYSTEM**

* Reduce a lot of time and effort
* Reduces the paper work
* Maintain the security
* Handle the data are easy
* Friendly user interface.

**3.5 APPLICATION OF SPECIFICATION**

Microsoft visual studio integrated development environment from Microsoft. It is used to develop computer program, as well as websites, web apps, web service, and mobile apps. Visual studio uses Microsoft software development platforms such as windows API, Windows presentation Foundation. windows store and Microsoft Silverlight. It can produce both native code and managed code.

**3.6 MODULE DESCRIPTION**

**STUDENT:**

* Student can give the feedback about the lectures on the scale of ten. Students can give manage about the lecturer based on

**STAFF:**

* The student management given by the student data viewed by the staff and improve their performance in teaching and other aspects.

**ADMIN:**

* Finally, report was referred by the admin and give counselling to lectures to improve their teaching.
* There are the three modules present in the project so we have developed these things in very easy and clearly understandable way.
* So, by seeing above description , the proposed system having many advantage as compared to the existing system.

**3.7 MODULE DESIGN**

A collection of data designed to be used by different people is called a database. It is a collection interrelated data stored together with controlled redundancy to serve one or more application in an optional fusion. The data is stored in such a fashion that is independent of the program of the people using the data. A common and controlled approach is used in adding new data and modifying and retrieving data with the database.

**STUDENT MODULE**

Login and provide the data about staffs.

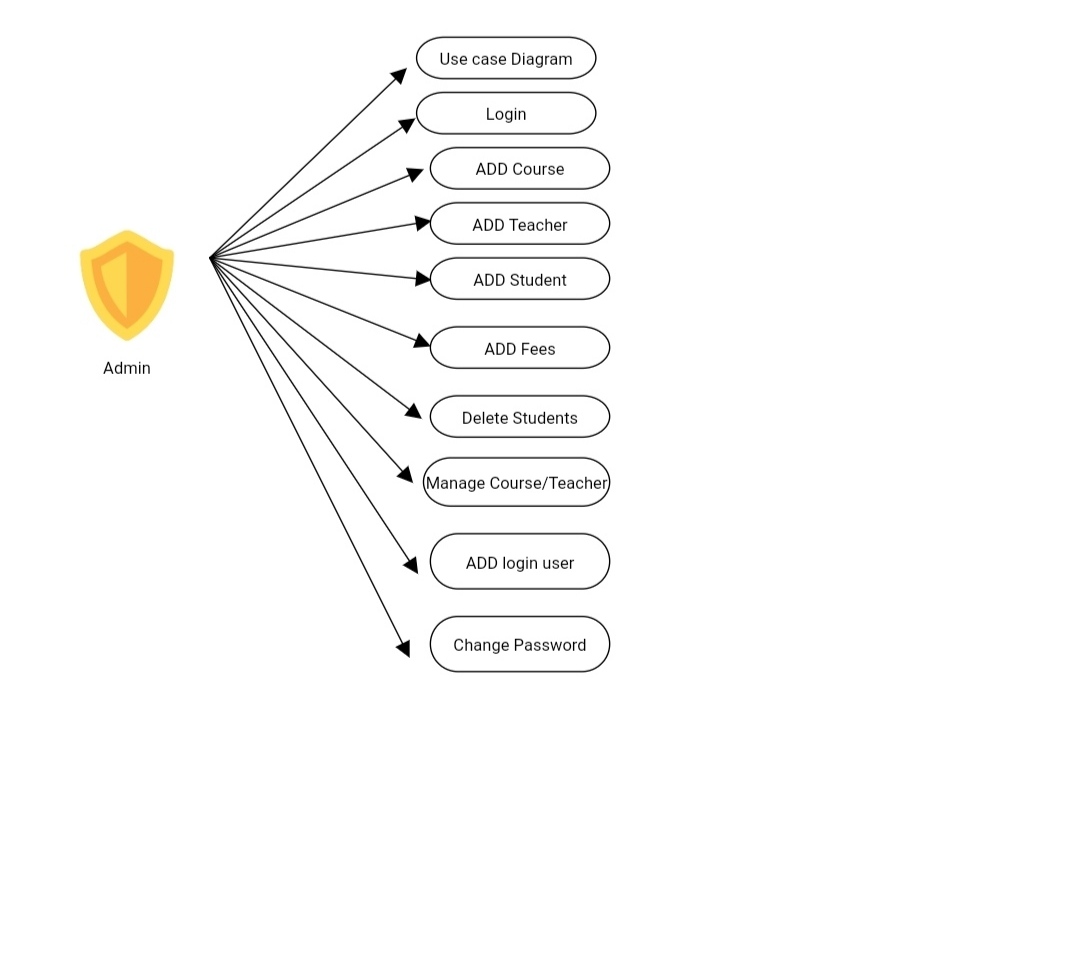
**STAFF MODULE**

View information provided by the students.

**ADMIN MODULE**

Refer data and maintain the records to the staff.

**3.8 BLOCK DIAGRAM**



3.8 Block Diagram

**CHAPTER -4**

**4.1 SYSTEM DESIGN**

System design is the solution to the creation of a new system. This phase is composed of several system. This phase focuses on the detailed implementation of the feasible system. It emphasis on training design specifications to performance specification. System design has two phases of development logical and physical design.

During design phase the analyst describes inputs , databases, and procedure all in format that means the uses requirement. The analyst also specifies the user needs and at a level that virtually determines the information flow into and out of the system and the data resources. Here the logical design is done through data flow diagrams and database design. The physical design is followed by physical design. Physical design produces the working system by defining the design specifications. Which tell the programmers exactly what the candidate system must do. The programmers write the necessary programs the accept input from the user, perform necessary processing on accepted data through call and produce the required report on a hard copy.

The physical design is followed by physical design. Physical design produces the working system by defining the design specifications. Which tell the programmer exactly what the candidate system must do. The programmers write the necessary program that accept input from the user, perform necessary processing on accepted data through call and produce the required report display on the screen.

**4.2 SOFTWARE REQUIREMENTS**

**OS:** Windows 10 or above

GOOGLE CHROME /INTERNET EXPLORER

XAMPP (version 3-7)

python main editor: PyCharm community

Workspace editor: Sublime text 3

**4.3 HARDWARE SPECIFICATION**

**COMPUTER: 1.1** GHZ

Minimum 2GB of RAM

2.5GB OF Available hard disk

5400 RPM Hard drive

DVD ROM drive

1366\*768 higher resolution display

**4.4 LIBRARIES DEVELOPMENT**

The STUDENT MANAGEMENT SYSTEM Class Libraries is a library of classes, interfaces and value types that provide access to system functionality. We can say that it is the foundation of the Framework on which the applications, controls and components of the, STUDENT Management are built. In .students many new APIs have been added to enable the following key scenario : Some Culture info properties are read write rather than read only.

**4.5 PROGRAMMING LANGUAGE**

STUDENT MANAGEMENT SYSTEM is an open source ,server side web application framework designed for web development to produce dynamic web pages. It was developed by Microsoft to allow programmers to build dynamic websites, applications and services. The name stands for Active server page Network Enabled Technologies. It was first released on January 2002 with version 1.0 of the Framework and is the successor to Microsoft Active Server page(ASP) technology is built on the common language Runtime (CLR) allowing programmers to write code using ant supported. STUDENT DETAILS components to process SOAP message. These applications have window authentication which makes the application more secure. In this applications source code and student files are together ,so student pages are easy to manage and write. The source code is now being run on the computer. This gives the websites a lot of strength and versatility.

**4.6 OPERATING SYSTEM SUPPORT**

Visual studio 2017 is supported in the following operating systems:

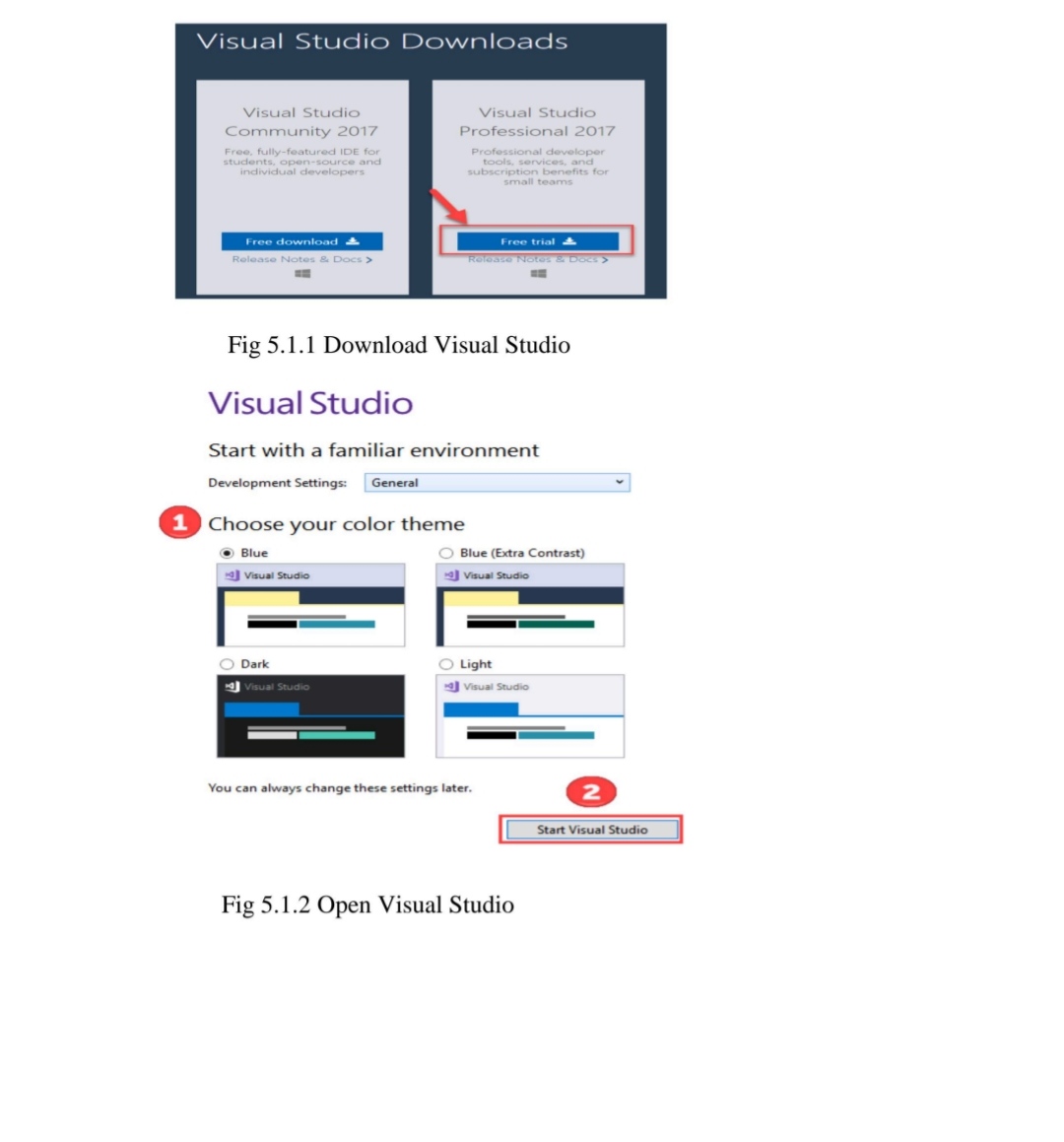
* Windows 10 or Higher ; Home, Education and enterprise.
* Windows Server 2016: Standard and Datacenter
* Windows 8.1: Core, Professional and Enterprise.
* Windows Server 2012 R2: Essentials, Standards, Datacenter, Ultimate.

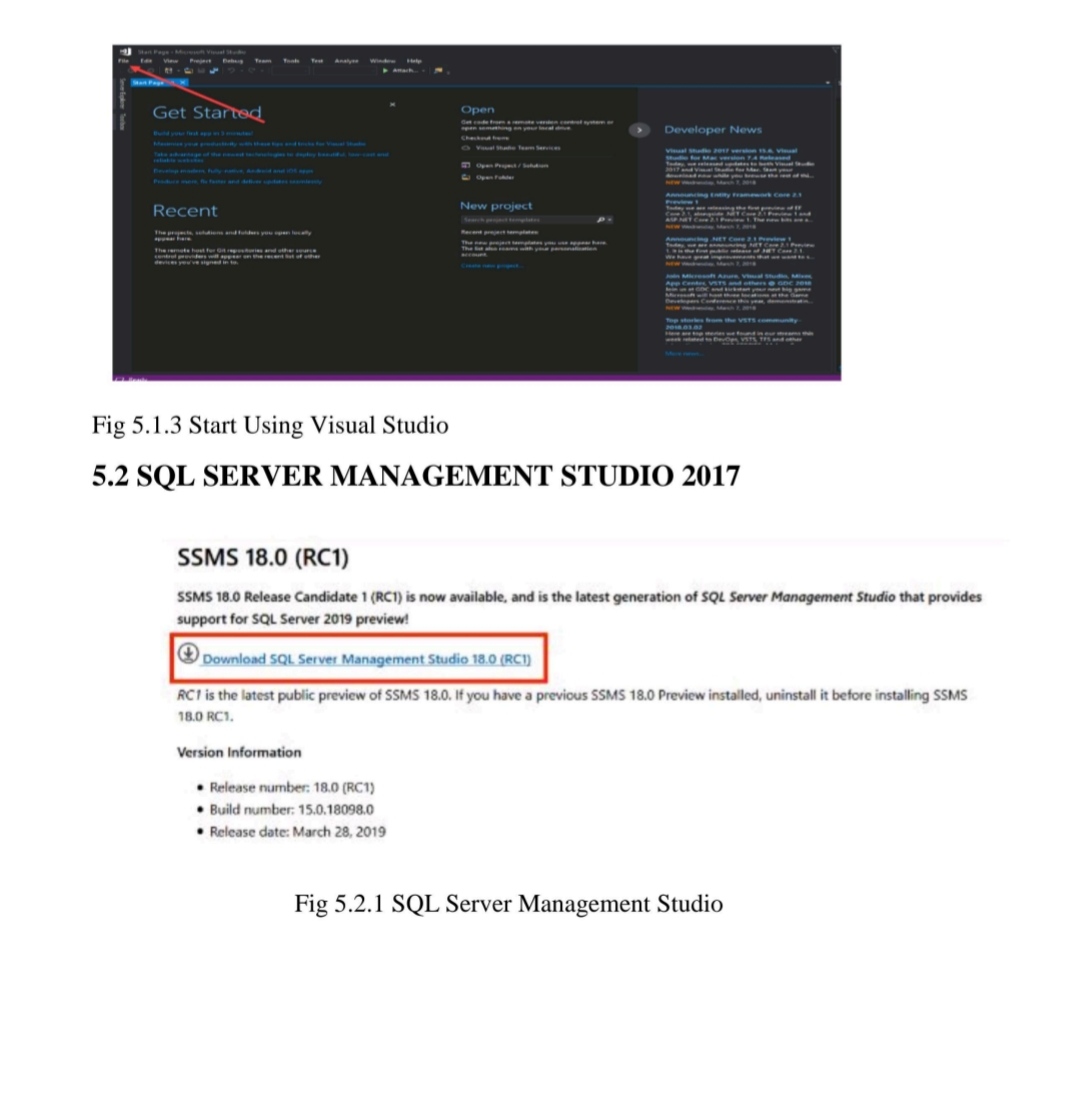
**CHAPTER 5**

**SYSTEM IMPLEMENTATION**

**5.1 INSTALLATION**

**VISUAL STUDIO 2017**

****

****

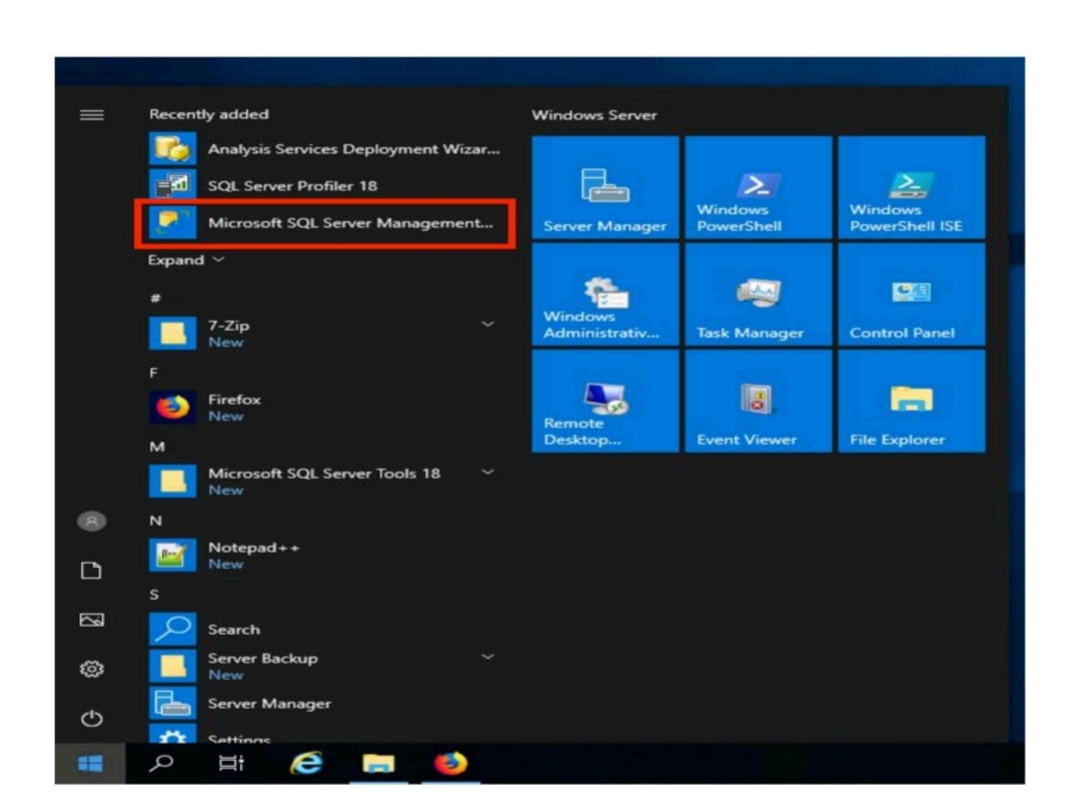


fig 5.2.2 Microsoft SQL Server Management Studio

**CHAPTER 6**

**SYSTEM TESTING**

**6.1 SYSTEM TESTING**

Testing is a set activity that can be planned and conducted systematically. Testing begins at the module level and work towards the integration of entire computer based system. Nothingis complete without testing, as it is vital success of the system.

Testing Objectives:

There are several rules that can serve as testing objectives, they are

1. Testing is a process of executing a program with the intent of finding an error.
2. A good test case is one that has high probability of finding an undiscovered error.
3. A successful test is one that uncovers an undiscovered error.

If testing is conducted successfully according to the objectives as stated above, it would uncover errors in the software. Also testing demonstrates that software functions appear to the working according to the specification, that performance requirements appear to have been met.

There are three ways to test a program

1. For Correctness.
2. For Implementation efficiency.
3. For Computational Complexity.

Test for correctness are supposed to verify that a program does exactly what it was designed to do. This is much more difficult than it may at first appear, especially for large programs.

Tests for implementation efficiency attempt to find ways to make a correct program faster use less storage. It is a code refining process, which reexamines the implementation phase of algorithm development.

Tests for computational complexity amount to an experimental analysis of the complexity of an algorithm or an experimental comparison of two or more algorithms, which solve the same problem.

Testing Correctness

The following ideas should be a part of any testing plan:

1. Preventive Measures
2. Spot checks
3. Testing all parts of the program
4. Test data
5. Looking for trouble
6. Time for testing
7. Re testing

The data is entered in all forms separately and whenever an error occurred, it is corrected immediately. A quality team deputed by the management verified all the necessary documents and tested the software while entering the data at all levels. The entire testing process can be divided into 3 phases

1. Unit Testing
2. Integrated Testing
3. System Testing

**6.1.1 UNIT TESTING**

As this system was partially GUI based WINDOWS application, the following were tested in this phase

1. Tab Order
2. Reverse Tab Order
3. Fie id length
4. Front end validations

In our system, unit testing has been successfully handled. The test data was given to each and every module in all aspects and got the desired output. Each module has been tested found working properly.

**6.1.2 INTEGRATION TESTING**

The data should be prepared carefully since the data only determines the efficiency and accuracy of the system. Artificial data are prepared solely for testing. Every program validates the input data. Each module has been tested found working properly.

**6.1.3 VALIDATION TESTING**

In this, all the code modules were tested individually one after the order.

The following were tested in all the modules,

1. Loop testing
2. Boundary Value Analysis
3. Equivalence Partitioning Testing

In our case all the modules were combined and given the test data. The combined module works successfully without any side effect on other programs. Everything was found fine working.

**6.1.4 OUTPUT TESTING**

This is the final step is testing. In this the entire system was tested as a whole with all forms, code, modules and class modules. This form of testing is popularly known as Black Box testing or system testing.

Black Box testing methods focus on the functional requirement of the software. That is, Black Box testing enables the software engineer to derive sets of input condition that will fully exercise all functional requirements for a program. Black Box testing attempts to find errors in the following categories; incorrect or missing function, interface errors, errors in data structure or external database access, performance errors and initialization errors and termination errors.

**CHAPTER 7**

**RESULT ANALYSIS AND OUTPUTS**

* 1. **LOGIN PAGE**

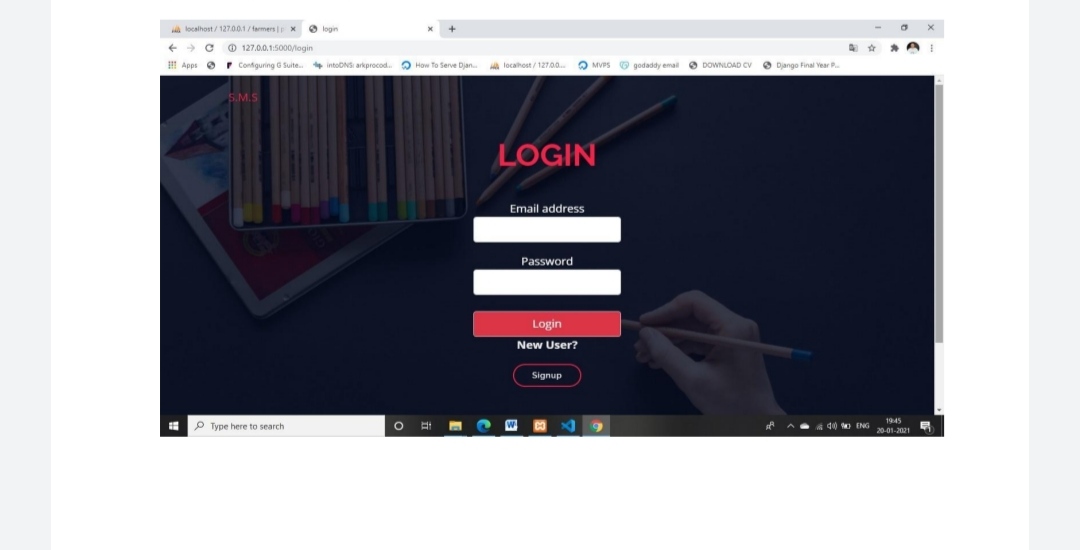
****

Fig 7.1 login page

In login page there are 3 modules such as Email address, Password, and login user. By clicking each module we direct to login page for each.

* 1. **ADD STUDENT INFO**

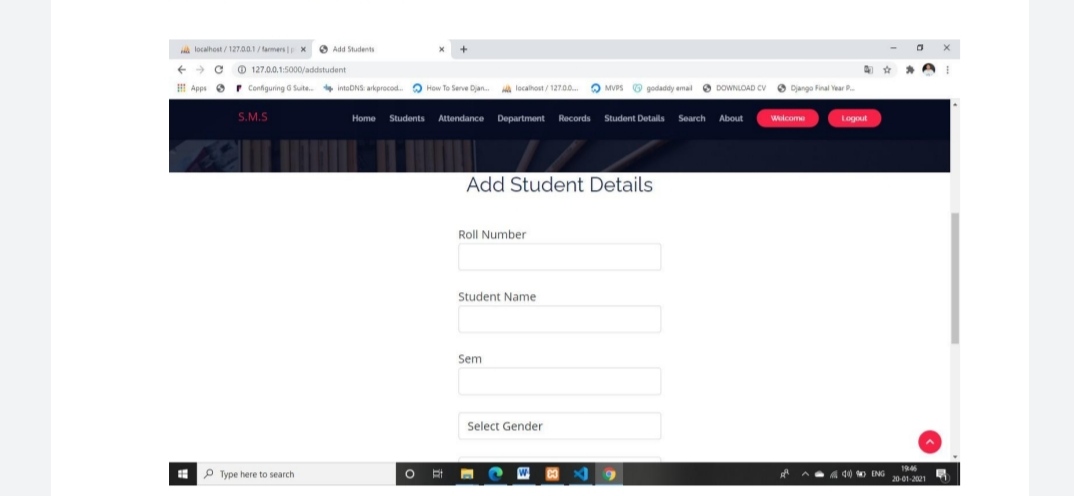


Fig 7.2 Add student info

Only the admin can add new student details and list of the students. And also only the admin have authority to remove the particular staff from the list, if they are not avail in the work. After collecting student details, admin can view the details given by students individually.

* 1. **TRIGGERS RECORDS**

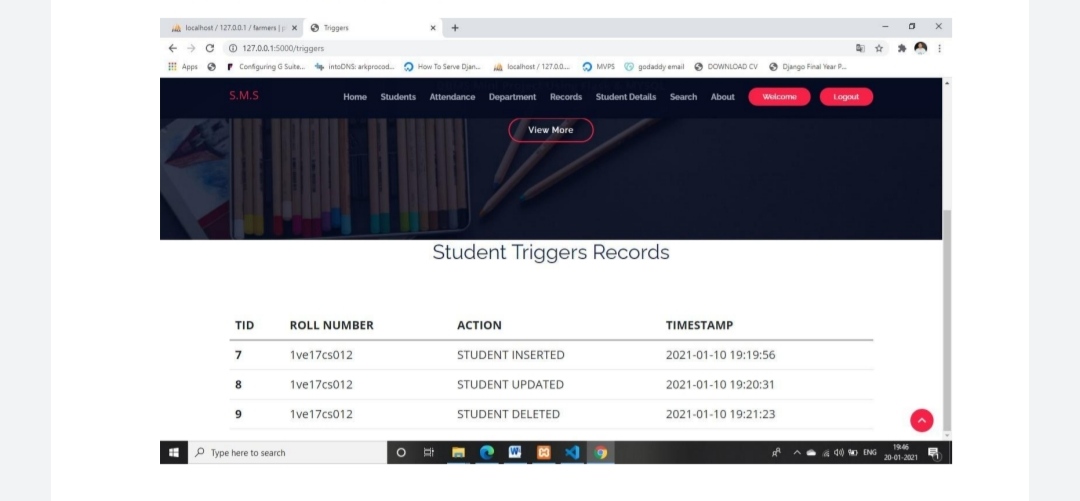
****

Fig 7.3 Triggers records

Student can view the details for each one and staff through their unique login. The ranges between 1-2.

**7.4 DATABASE LOCALHOST**

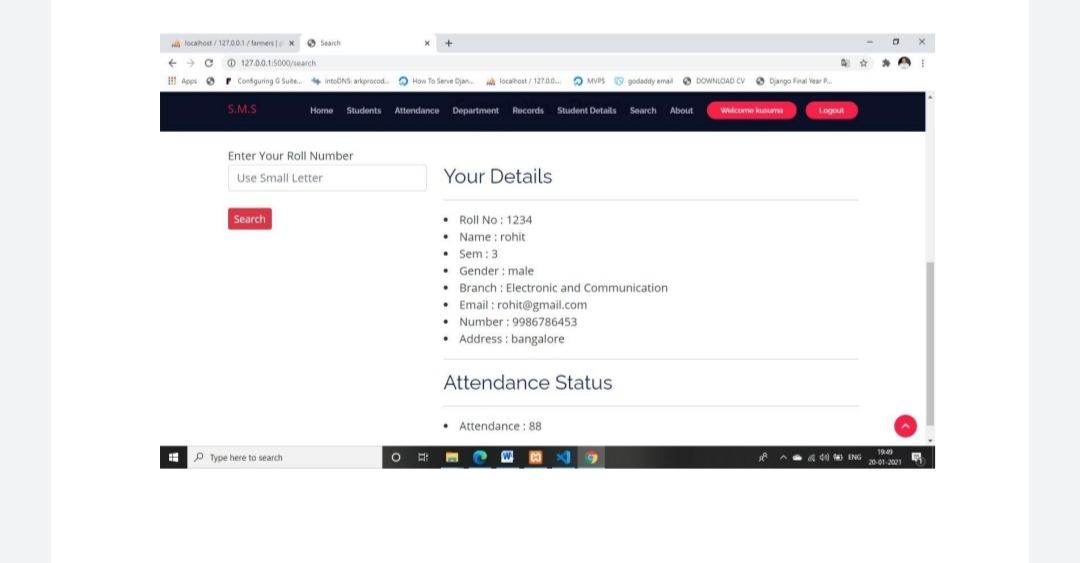
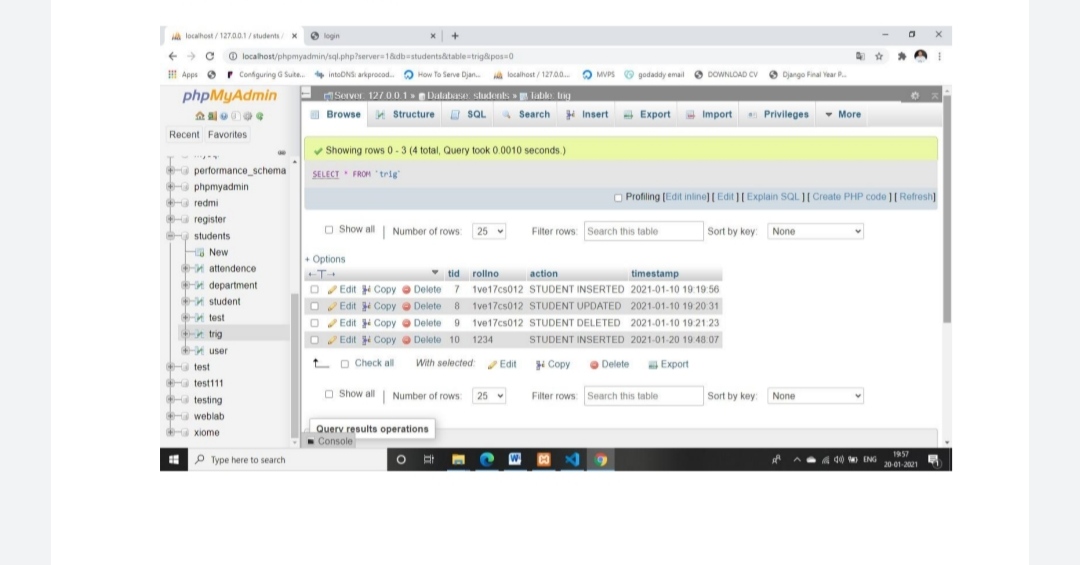
****

Fig 7.4 Database localhost

The student details can collect and maintained into excel sheets and stored into the database.



Staff can view the student details given by these system through their login with the hidden information of students.

**CHAPTER 8**

**CONCLUSION AND FUTURE ENHANCEMENT**

**8.1 CONCLUSION**

This project is designed for the purpose to reduce the staff time and to reduce the burden of maintaining huge amount of records of 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 manipulating according to the requirements with basic knowledge of SQL. As the comparison of manual record with existing system, the new proposed system is easier way to manage whole things in a particular manner.

**8.2 FUTURE ENHANCEMENT**

* Enhanced database storage facility.
* Enhanced user friendly GUI.
* More advanced results systems.
* Online feedback forms.

Various other captions can also be added for the better usability of project.

**APPENDIX**

**Coding**

**Home page: Default. aspx. cs**

## D 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 Attendence(db.Model): aid=db.Column(db.Integer,primary\_key=True) rollno=db.Column(db.String(100)) attendance=db.Column(db.Integer())

class Trig(db.Model): tid=db.Column(db.Integer,primary\_key=True) rollno=db.Column(db.String(100)) action=db.Column(db.String(100)) timestamp=db.Column(db.String(100))

class User(UserMixin,db.Model): id=db.Column(db.Integer,primary\_key=True) username=db.Column(db.String(50)) email=db.Column(db.String(50),unique=True) password=db.Column(db.String(1000))

class Student(db.Model): id=db.Column(db.Integer,primary\_key=True) rollno=db.Column(db.String(50)) sname=db.Column(db.String(50)) sem=db.Column(db.Integer) gender=db.Column(db.String(50)) branch=db.Column(db.String(50)) email=db.Column(db.String(50)) number=db.Column(db.String(12)) address=db.Column(db.String(100))

@app.route('/') def index():

return render\_template('index.html')

@app.route('/studentdetails') def studentdetails():

query=db.engine.execute(f"SELECT \* FROM `student`") return render\_template('studentdetails.html',query=query)

@app.route('/triggers') def triggers():

query=db.engine.execute(f"SELECT \* FROM `trig`") return render\_template('triggers.html',query=query)

@app.route('/department',methods=['POST','GET']) def department():

if request.method=="POST": dept=request.form.get('dept') query=Department.query.filter\_by(branch=dept).first() if query:

flash("Department Already Exist","warning") return redirect('/department')

dep=Department(branch=dept) db.session.add(dep) db.session.commit() flash("Department Addes","success")

return render\_template('department.html')

@app.route('/addattendance',methods=['POST','GET']) def addattendance():

query=db.engine.execute(f"SELECT \* FROM `student`") if request.method=="POST":

rollno=request.form.get('rollno') attend=request.form.get('attend') print(attend,rollno) atte=Attendence(rollno=rollno,attendance=attend) db.session.add(atte)

db.session.commit()

flash("Attendance added","warning")

return render\_template('attendance.html',query=query)

@app.route('/search',methods=['POST','GET']) def search():

if request.method=="POST": rollno=request.form.get('roll') bio=Student.query.filter\_by(rollno=rollno).first() attend=Attendence.query.filter\_by(rollno=rollno).first()

return render\_template('search.html',bio=bio,attend=attend)

return render\_template('search.html')

@app.route("/delete/<string:id>",methods=['POST','GET']) @login\_required

def delete(id):

db.engine.execute(f"DELETE FROM `student` WHERE `student`.`id`={id}") flash("Slot Deleted Successful","danger")

return redirect('/studentdetails')

@app.route("/edit/<string:id>",methods=['POST','GET']) @login\_required

def edit(id):

dept=db.engine.execute("SELECT \* FROM `department`") posts=Student.query.filter\_by(id=id).first()

if request.method=="POST": rollno=request.form.get('rollno') sname=request.form.get('sname') sem=request.form.get('sem') gender=request.form.get('gender') branch=request.form.get('branch') email=request.form.get('email') num=request.form.get('num') address=request.form.get('address') query=db.engine.execute(f"UPDATE `student` SET

`rollno`='{rollno}',`sname`='{sname}',`sem`='{sem}',`gender`='{gender}',`branch`='{branch}',`email`='{em ail}',`number`='{num}',`address`='{address}'")

flash("Slot is Updates","success") return redirect('/studentdetails')

return render\_template('edit.html',posts=posts,dept=dept)

@app.route('/signup',methods=['POST','GET']) def signup():

if request.method == "POST": username=request.form.get('username') email=request.form.get('email') password=request.form.get('password') user=User.query.filter\_by(email=email).first() if user:

flash("Email Already Exist","warning") return render\_template('/signup.html')

encpassword=generate\_password\_hash(password)

new\_user=db.engine.execute(f"INSERT INTO `user` (`username`,`email`,`password`) VALUES ('{username}','{email}','{encpassword}')")

# this is method 2 to save data in db

# newuser=User(username=username,email=email,password=encpassword) # db.session.add(newuser)

# db.session.commit()

flash("Signup Succes Please Login","success") return render\_template('login.html')

return render\_template('signup.html')

@app.route('/login',methods=['POST','GET']) def login():

if request.method == "POST": email=request.form.get('email') password=request.form.get('password') user=User.query.filter\_by(email=email).first()

if user and check\_password\_hash(user.password,password): login\_user(user)

flash("Login Success","primary") return redirect(url\_for('index'))

else:

flash("invalid credentials","danger") return render\_template('login.html')

return render\_template('login.html')

@app.route('/logout') @login\_required

def logout(): logout\_user()

flash("Logout SuccessFul","warning") return redirect(url\_for('login'))

@app.route('/addstudent',methods=['POST','GET']) @login\_required

def addstudent():

dept=db.engine.execute("SELECT \* FROM `department`") if request.method=="POST":

rollno=request.form.get('rollno') sname=request.form.get('sname') sem=request.form.get('sem') gender=request.form.get('gender') branch=request.form.get('branch') email=request.form.get('email') num=request.form.get('num') address=request.form.get('address') query=db.engine.execute(f"INSERT INTO `student`

(`rollno`,`sname`,`sem`,`gender`,`branch`,`email`,`number`,`address`) VALUES ('{rollno}','{sname}','{sem}','{gender}','{branch}','{email}','{num}','{address}')")

flash("Booking Confirmed","info")

return render\_template('student.html',dept=dept) @app.route('/test')

def test(): try:

Test.query.all()

return 'My database is Connected' except:

return 'My db is not Connected'

app.run(debug=True)

**REFERENCES**

[1] S.R. Bharamagoudar, “web based student information management system,” International journal of advanced Research in computer and communication Engineering. Issue 6 june 2013

[2] Kartiki Datakar , “ONLINE COLLGE MANAGEMENT SYSTEM” International journal of computer science and mobile computing,Issue 4 april 2016.

[3] Lalit joshi “ A RESEARCH PAPER ON COLLEGE MANAGEMENT SYSTEM,” International journal of computer applications Issue 11,July 2015.

[4] Isbudeen Noor Mohamed “ STUDENT INFORMATION SYSTEM WITH SMS, International conference on computer communication ,jan07 ,2016.

[4] <https://github.com/android>

[5] <https://php.net/>

[6] [https://www.mysqltutorial.org/mysql resources.aspx](https://www.mysqltutorial.org/mysql%20resources.aspx)

[7] <https://www.tutorialspoint.com/mysql>

[8] pressman, Roger S., Software engineering ,fifth edition ,McGraw-Hill Higher education,2001.

[9] Elmasri | Navathe Fundamentals of Database systems ,third edition Addison -wesley ,2000

[10] Beginning ASP.NET 4.0 in c# by Mathew MacDonald.