

STUDENTS PORTAL

A mini Project Report

Submitted for partial fulfilment for the award of the degree of

BACHELOR OF COMPUTER APPLICATIONS

By

M.VISHAL (A21CADB60)

R.SHANMUGA SUNDHARAM (A21CADB52)

M.MOHAN RAJ (A21CADB36)

Under the Guidance of

Dr.A.Lourdu Caroline, M.C.A., M.Phil., Ph.D.,

(Asst. Professor, Post Graduate Department of Computer Applications)



PG DEPARTMENT OF COMPUTER APPLICATIONS

ST. JOSEPH'S COLLEGE OF ARTS AND SCIENCE

(AUTONOMOUS)

AFFILIATED TO ANNAMALAI UNIVERSITY, CHIDAMBARAM

CUDDALORE-607001

APRIL-2024

CERTIFICATE

This is to certify that the mini project entitled

STUDENTS PORTAL

Being Submitted to the St. Joseph's College of Arts and Science (Autonomous),

Cuddalore-1

(Affiliated to Annamalai University, Chidambaram)

By

M.VISHAL (A21CADB60)

R.SHANMUGA SUNDHARAM (A21CADB52)

M.MOHAN RAJ (A21CADB36)

For the partial fulfillment for the award of degree of

BACHELOR OF COMPUTER APPLICATIONS

is a bonafide record of work carried out by them,

Under my guidance and supervision.

INTERNAL GUIDE

HEAD OF THE DEPARTMENT

Submitted for the viva-voce examination held on_____

Examiners:

1._____

2._____

ACKNOWLEDGEMENT

First and Fore most, We thank God, the Almighty, for the showering his kindness and blessings to complete this project successfully.

We extend our sincere gratitude to respected **Rev.Fr.M.Swaminathan D.C.L., D.JURIS.**, Secretary and **Dr.M.Arumai Selvam M.Sc.(Maths), M.Sc.(CS), M.Phil.(CS), Ph.D.**, Principal of St.Joseph's College of Arts & science (Autonomous) for providing such a good congenial environment to enlighten our knowledge.

We extend our hearty gratitude to **Dr.A.John Pradeep Ebenezer., M.C.A., M.Phil., Ph.D.**, Head of PG and Department of Computer Applications for the moral support and encouragement during the project.

We extend our hearty gratitude to our guide **Dr.A.Lourdu Caroline., M.C.A., M.Phil., Ph.D.**, Assistant professor of PG Department of Computer Applications for the guidance during the project.

At the Outset We take Pleasure to extend our sincere gratitude to all our Teachers who had taken pains to shape and model us in all our endeavors. We bound to express our sincere and hearty gratitude to our dear parents for their financial and material support which helped us to make this project a successful one. Last but not the Least We place deepest sense of gratitude us to all our friends who has supported to complete this project more successfully.

M.VISHAL

R.SHANMUGA SUNDHARAM

M.MOHAN RAJ

TABLE OF CONTENT

S. NO	DESCRIPTION	PAGE NO
1	ABSTRACT	1
2	OVERVIEW OF THE PROJECT	2
3	SYSTEM STUDY AND ANALYSIS	5
4	SYSTEM SPECIFICATION	8
5	SOFTWARE DISCRIPTION	15
6	SYSTEM IMPLEMENTATIONS	19
7	TESTING	45
8	APPENDIX	47
9	CONCLUSIONS AND FUTURE ENHANCEMENT	51
10	REFERENCES	52

1. ABSTRACT

This abstract offers a comprehensive overview of Student's Portal (ERP) systems, emphasizing their pivotal role in streamlining educational management processes within academic institutions. In an era of technological advancements and evolving educational paradigms, the effective integration of ERP systems has emerged as a transformative solution for optimizing administrative workflows, enhancing data-driven decision-making, and fostering student success.

Student ERP systems encompass a suite of integrated software modules designed to manage various aspects of student information, academic operations, and institutional resources. These modules typically include functionalities for student academic records management, course scheduling, financial aid administration, and among others. These advancements hold the promise of further enhancing the functionality and accessibility of ERP systems, thereby enabling institutions to adapt to evolving educational needs and trends. It is designed to replace the problem of no due form as the admin can view the pending amount of particular student and other details with no paper work.

In conclusion, this abstract underscores the transformative potential of Student ERP systems in revolutionizing educational management practices. By leveraging the capabilities of ERP technology, academic institutions can optimize resource allocation, improve administrative efficiency, and ultimately enhance the educational experience for students, faculty, and staff alike.

2. OVERVIEW OF THE PROJECT

- **INTRODUCTION**

Students Portal

ERP systems integrate various administrative functions such as student details, academic information, exam scores, attendance and other student informations. This integration streamlines administrative processes, reduces manual tasks, and eliminates redundant data entry, leading to improved efficiency and productivity. ERP systems integrate various administrative functions such as admissions, registration, student records, finance, human resources, and academic affairs into a single platform. This integration streamlines administrative processes, reduces manual tasks, and eliminates redundant data entry, leading to improved efficiency and productivity. ERP systems provide the flexibility to support evolving requirements and scale with the institution's growth.

Overall, implementing an ERP system in a college setting can significantly improve operational efficiency, enhance the student experience, and support the institution's mission of providing quality education and academic excellence.

This project is modularized as the following:

- **Admin Login**
- **Student Login**
- **Student Details**
- **Student's tutor details**
- **Attendance details**
- **Academic details**
- **Fee details**
- **Library details**
- **Hostel details**

1. Admin Login

This is one side of the program with a separate website for Admin to create and manage database. From here, all the student information will be added to the database which will enable student to access and view their details. It includes forms such as student detail, tutor detail, lab details, attendance, fee details, library and hostel detail.

The Admin can login to the website using his username and password and select the needed form to enter details and save it to the database.

2. Student's Login

It's the another side of program for the student to view their details of them. Each student will have unique username and password by which they can login to their respective ID through the wesbite and view the information such as student details, tutor details, lab details, attendance, fee details, library and hostel details. The details will be retrived from the database in which the admin has stored the datas previously.

3. Student Details

Student details form consist of the general details of the student such as register number, name, course, college mail ID, personal mail ID, Date of Birth, Mobile number, blood group and all other basic information about the students

4. Tutor Details

This consist the details of the Tutor of the student who is incharge of them. It will display the Tutor name, Tutor mail ID, Mobile number and their room number.It will help the student to get the contact details of the Staff incharge

5. Class Details

This form consist of the Class details and lab details of the student. It will display according to the course the student persues. This will help the student navigate which class and lab is assigned to him/her.

6. Attendance Details

Attendance form consist of the attendance of the student. It will display semester wise attendance of the student separately from Semester 1 to Semester 6 repectively. It will help the student to keep an eye on their attendance.

7. Academic Details

The academic form contains the marks of the students of each semester. It will display the GPA of the scale 0.0 to 10.0 from semester 1 to semester 6.This will help the student to calculate his/her CGPA and analyze the Overall Grade.

8. Fee Details

This Fee detail form displays the College Fee and Examination fee of the student It consist of separate column displaying Total amount, Amount paid, Amount to be paid, Exam fee amount, Exam fee paid and pending amount with their due dates respectively.

9. Library Details

Library details form consist of the details related to Library such as Total number of books borrowed from the library, Number of books returned and books remaining to be returned with their due dates, Fine amount if date over-due, Fine paid and remaining to be paid. It will help them to keep an account on books when to return the books without fine etc...

10. Hostel Details

This form consist of the details related to the hostel and mess of the students.It has information such as Hostel room number, Warden's name, Mess fees amount and its Paid status and its Fine amount if not paid respectively.

3. SYSTEM STUDY AND ANALYSIS

INTRODUCTION

System analysis is a process of gathering and interpreting facts, diagnosing problems and the information to recommend improvements on the system. It is a problem solving activity that requires intensive communication between the system users and system developers. System analysis or study is an important phase of any system development process. The system is studied to the minutest detail and analyzed. The system analyst plays the role of the interrogator and dwells deep into the working of the present system. The system is viewed as a whole and the input to the system are identified. The outputs from the organizations are traced to the various processes. System analysis is concerned with becoming aware of the problem, identifying the relevant and decisional variables, analyzing and synthesizing the various factors and determining an optimal or at least a satisfactory solution or program of action.

A detailed study of the process must be made by various techniques like interviews, questionnaires etc. The data collected by these sources must be scrutinized to arrive to a conclusion. The conclusion is an understanding of how the system functions. This system is called the existing system. Now the existing system is subjected to close study and problem areas are identified. The designer now functions as a problem solver and tries to sort out the difficulties that the enterprise faces. The solutions are given as proposals. The proposal is then weighed with the existing system analytically and the best one is selected. The proposal is presented to the user for an endorsement by the user. The proposal is reviewed on user request and suitable changes are made. This is loop that ends as soon as the user is satisfied with proposal.

Preliminary study is the process of gathering and interpreting facts, using the information for further studies on the system. Preliminary study is problem solving activity that requires intensive communication between the system users and system developers. It does various feasibility studies. In these studies a rough

figure of the system activities can be obtained, from which the decision about the strategies to be followed for effective system study and analysis can be taken.

Here in the project Students Portal, a detailed study of existing system is carried along with all the steps in system analysis. An idea for creating a better project was carried and the next steps were followed.

FEASIBILITY STUDY

An important outcome of the preliminary investigation is the determination that the system requested is feasible. Feasibility study is carried out to select the best system that meets the performance requirements.

Feasibility study is both necessary and prudent to evaluate the feasibility of the project at the earliest possible time. It involves preliminary investigation of the project and examines whether the designed system will be useful to the organization. Months or years of effort, thousand for millions of money and untold professional embarrassment can be averted if an in-conceived system is recognized early in the definition phase.

The different types of feasibility are: Technical feasibility, Operational feasibility, Economical feasibility.

TECHNICAL FEASIBILITY

Technical Feasibility deals with the hardware as well as software requirements. Technology is not a constraint to type system development. We have to find out whether the necessary technology, the proposed equipments have the capacity to hold the data, which is used in the project, should be checked to carry-out this technical feasibility.

The technical feasibility issues usually raised during the feasibility stage of investigation includes these:

- This software is running in windows 11 Operating System, which can be easily installed.
- The hardware required is Intel based.
- The system can be expanded.

BEHAVIORAL FEASIBILITY

This feasibility test asks if the system will work when it is developed and installed.

Operational feasibility in this project:

- The proposed system offers greater level of user-friendliness.
- The proposed system produces best results and gives high performance. It can be implemented easily .So this project is operationally feasible.

ECONOMICAL FEASIBILITY

Economical Feasibility deals about the economical impact faced by the organization to implement a new system. Financial benefits must equal or exceed the costs. The cost of conducting a full system, including software and hardware cost for the class of application being considered should be evaluated. Economic Feasibility in this project:

- The cost to conduct a full system investigation is possible.
- There is no additional manpower requirement.
- There is no additional cost involved in maintaining the proposed system.

4. SYSTEM SPECIFICATION

Hardware Specification

Processor	: 12th Gen Intel(R) Core(TM) i5-1240P
Clock speed	: 4.4GHz
System bus	: 64 bits
RAM	: 8GB
SSD	: 512GB
Laptop	: HP Pavilion 15

Software Specification

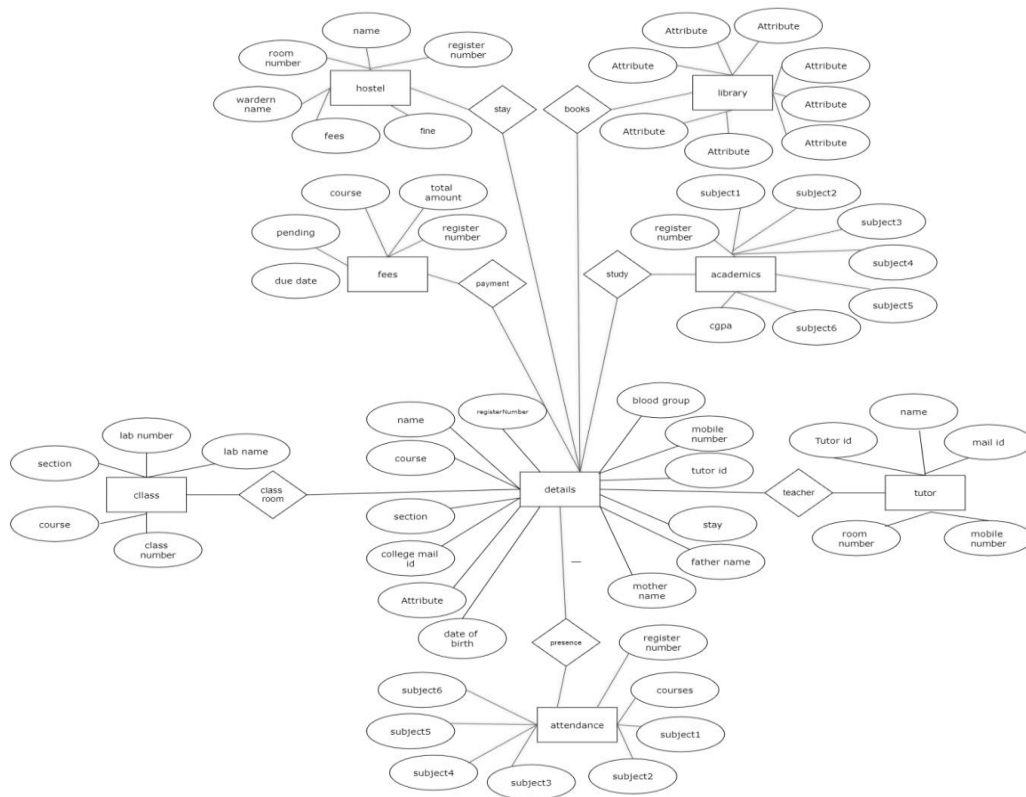
OS	:	MS WINDOWS 11
Front End	:	HTML5, CSS3
Back End	:	PYTHON IDLE
Framework	:	FLASK
Database	:	SQLITE (DB BROWSER)

DATA FLOW DIAGRAM

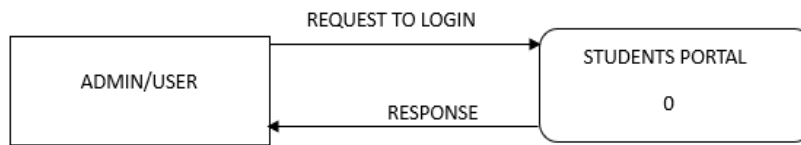
Data flow oriented techniques advocate that the major data items handled by a system must be first identified and then the processing required on these data items to produce the desired outputs should be determined. The DFD (also called as bubble chart) is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on these data, and the output generated by the system. It was introduced by De Macro (1978), Gane and Sarson (1979).

Data Flow Diagram

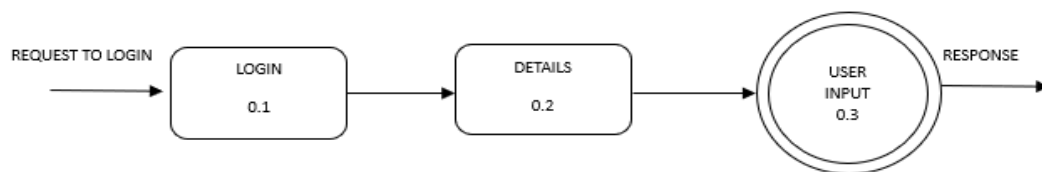
Context Diagram:



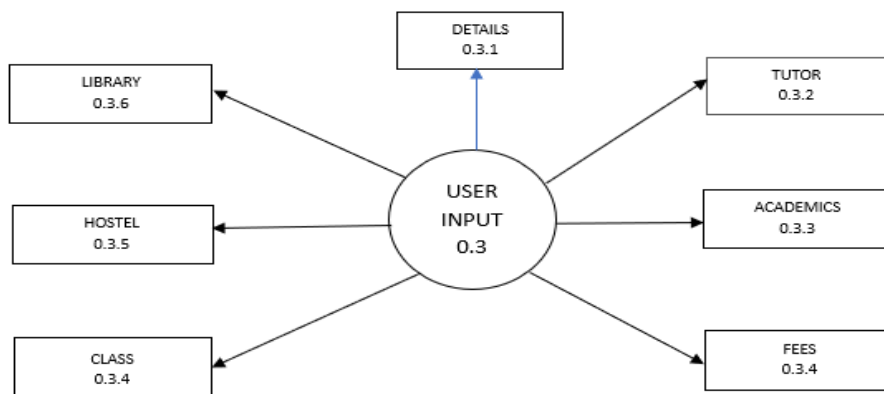
Level 1:



Level 2:



Level 3:



DATABASE DESIGN

A database is an organized mechanism that has the capability of storing information through which a user can retrieve stored information in an effective and efficient manner. The data is the purpose of any database and must be protected. Here, SQLite Database is used to store data.

The database design is a two-level process. In the first step, user requirements are gathered together and a database is designed which will meet these requirements as clearly as possible. This step is called Information Level Design and it is taken independent of any individual Database Management System (DBMS).

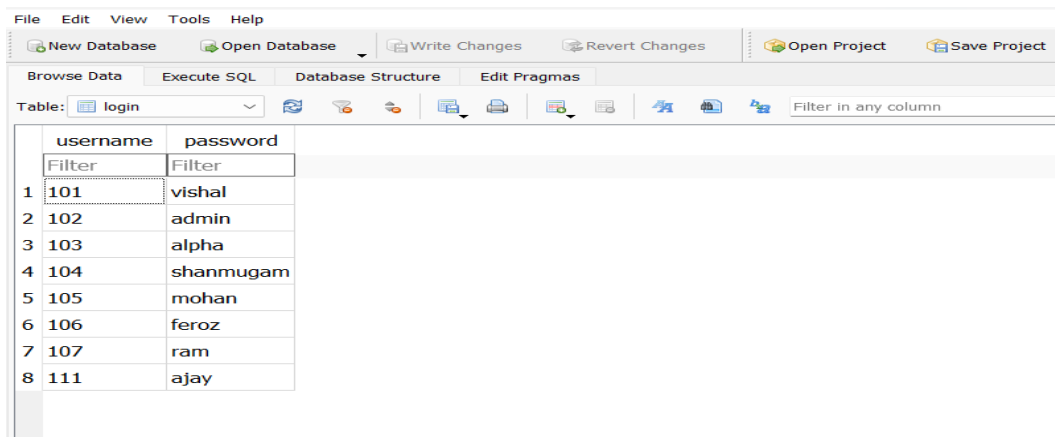
In the second step, this Information level design is transferred into a design for the specific DBMS that will be used to implement the system in question. This step is called Physical Level Design, concerned with the characteristics of the specific DBMS that will be used. A database design runs parallel with the system design. The organization of the data in the database is aimed to achieve the following two major objectives:

- Data Integrity
- Data independence

DATABASE TABLE DESIGN

Table Name: - Login

Primary key: username



The screenshot shows a database application interface. At the top, there is a menu bar with 'File', 'Edit', 'View', 'Tools', and 'Help'. Below the menu bar is a toolbar with buttons for 'New Database', 'Open Database', 'Write Changes', 'Revert Changes', 'Open Project', and 'Save Project'. Below the toolbar is a tabbed interface with 'Browse Data', 'Execute SQL', 'Database Structure', and 'Edit Pragmas'. The 'Browse Data' tab is active, showing a table named 'login'. The table has two columns: 'username' and 'password'. The 'username' column is highlighted, indicating it is the primary key. The table contains 8 rows of data, with the first row being a header row and the subsequent 7 rows being data rows. The data rows are numbered 1 through 8 in the first column.

	username	password
	Filter	Filter
1	101	vishal
2	102	admin
3	103	alpha
4	104	shanmugam
5	105	mohan
6	106	feroz
7	107	ram
8	111	ajay

Table Name: - Details

Primary Key: Id

FileEditViewToolsHelp

New DatabaseOpen DatabaseWrite ChangesRevert ChangesOpen ProjectSave Project

Browse DataExecute SQLDatabase StructureEdit Pragmas

Table: detailsFilter in any column

id	registerno	name	course	Print currently browsed table data [Ctrl+F]	smid	byear	eyear	dob	bloodgroup	mobilen	aadhar	fathername	mothername	parentmobilen	resiaddress
Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	101	101 Vishal	BCA	B	s101@gmail.com vishal@gmail.com	2021	2024	14-07-2004	B+	9876543210	123046568416	Mukesh	Rekha	6666460965	123,xyz street,abc
2	102	102 Admin	BCA	B	s102@gmail.com admin@gmail.com	2021	2024	01-01-2004	AB+	8776453454	297378517551	Mohan	Mohani	9637418520	0.abc street,xyz
3	103	103 Alpha	BCA	A	s103@gmail.com alpha@gmail.com	2021	2024	12-02-2003	A1+	9464418912	987451568256	Bala	Ramya	9873210785	53,norway street,new york
4	104	104 Shanmugam	BCA	B	s104@gmail.com somu@gmail.com	2021	2024	08-07-2004	A1+	8525857998	454789432672	Ram	Tamizh	9638527418	37,north street,Mampattu
5	105	105 Mohan	BCA	B	s105@gmail.com mohan@gmail.com	2021	2024	30-04-2004	O+	9344325153	753142785319	Muthu	Lakshmi	6565878341	3A,Balaji nagar,Cuddalore
6	106	106 Feroz	BCA	B	s106@gmail.com feroz@gmail.com	2021	2024	2004-06-03	O+	8248061797	79654123321	Ali Khan	Nisha	9630124578	Any Street,College backside,Cuddalore
7	107	111 Ajay	Computer Science	B	s111@gmail.com ajay@gmail.com	2023	2026	2006-06-07	AB-	9966332211	175498641245	Arun	Aarthi	9855441376	1,Annamalal nagar,Cuddalore

Table Name: - Academics

FileEditViewToolsHelp

New DatabaseOpen DatabaseWrite ChangesRevert ChangesOpen ProjectSave Project

Browse DataExecute SQLDatabase StructureEdit Pragmas

Table: academicsFilter in any column

	id	registerno	sub1	sub2	sub3	sub4	sub5	sub6	cgpa
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	101	101	6.2	6.8	6.4	6.0	5.0	7.2	6.2
2	102	102	9.0	6.7	7.1	5.8	6.7	5.9	6.4
3	103	103	8.7	6.9	6.9	8.1	6.4	7.0	6.2
4	104	104	8.4	6.7	6.9	5.7	6.8	7.2	7.6
5	105	105	6.1	6.2	6.3	6.4	6.5	5.6	6
6	106	106	7.2	7.4	6.9	9.2	7.4	8.6	7.9
7	111	111	6.2	4.6	5.9	6.7	5.8	6.4	5.9

Table Name: - Attendance

FileEditViewToolsHelp

New DatabaseOpen DatabaseWrite ChangesRevert ChangesOpen ProjectSave Project

Browse DataExecute SQLDatabase StructureEdit Pragmas

Table: attendanceFilter in any column

	id	registerno	course	sub1	sub2	sub3	sub4	sub5	sub6
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	101	101	BCA	99	98	97	99	99	97
2	102	102	BCA	73	83	79	96	92	97
3	103	103	BCA	69	64	71	56	64	75
4	104	104	BCA	73	81	76	83	76	76
5	105	105	BCA	45	83	76	49	55	71
6	106	106	BCA	89	75	87	46	99	79
7	111	111	Computer Science	75	46	86	79	89	99

Table Name: - Class

File Edit View Tools Help

New Database Open Database Write Changes Revert Changes Open Project Save Project

Browse Data Execute SQL Database Structure Edit Pragmas

Table: class Filter in any column

	cno	course	sec	labno1	labname1	labno2	labname2	labno3	labname3
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	
1	LAB01	BCA	B	IT01	PYTHON	CS03	PHP	IT02	JAVA
2	LAB02	BCA	A	IT01	PYTHON	IT01	C#	CS01	R
3	LAB03	BCA	C	IT01	PYTHON	CS03	PHP	CS02	.NET

Table Name: - Fees

File Edit View Tools Help

New DatabaseOpen DatabaseWrite ChangesRevert ChangesOpen ProjectSave Project

Browse DataExecute SQLDatabase StructureEdit Pragmas

Table: feesFilter in any column

	id	registerno	course	tamt	tpaid	tpending	tduedate	eamt	epaid	epending	eduedate
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	101	101	BCA	50000	50000	0	2023-02-18 23:59:59	2000	2000	0	2023-06-22 23:59:59
2	102	102	BCA	50000	35000	15000	2023-02-18 23:59:59	2000	0	2000	2023-06-22 23:59:59
3	103	103	BCA	50000	40000	10000	2023-02-18 23:59:59	2000	1500	500	2023-06-22 23:59:59
4	104	104	BCA	50000	15000	35000	2023-02-18 23:59:59	2000	1000	1000	2023-06-22 23:59:59
5	105	105	BCA	50000	15000	35000	2023-02-18 23:59:59	2000	1000	1000	2023-06-22 23:59:59
6	106	106	BCA	50000	20000	30000	2025-01-31 23:59:59.000000	2000	2000	0	2025-01-31 23:59:59
7	111	111	Computer Science	45000	30000	15000	2025-05-12 23:59:59.000000	1200	1200	0	2025-01-31 23:59:59

Table Name: - Hostel

File Edit View Tools Help

New DatabaseOpen DatabaseWrite ChangesRevert ChangesOpen ProjectSave Project

Browse DataExecute SQLDatabase StructureEdit Pragmas

Table: hostelFilter in any column

	id	registerno	name	roomno	wname	mfees	paidstatus	fine
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	101	101	Vishal	100	DAVID	2050	YES	0
2	102	102	Admin	110	DAVID	2250	NO	200
3	103	103	Alpha	123	ORTON	2500	NO	100
4	104	104	Shanmugam	169	DAVID	2000	YES	0
5	105	105	Mohan	102	DAVID	3000	NO	500
6	106	106	Feroz	111	Anonymous	2750	Yes	0
7	111	111	Ajay	46	Alex	4600	Yes	200

Table Name: - Fees

FileEditViewToolsHelp

New Database

Open Database

Write Changes

Revert Changes

Open Project

Save Project

Browse Data

Execute SQL

Database Structure

Edit Pragmas

Table: library

Filter in any column

	id	registerno	borrowedb	returnedb	reservedb	totalf	paid	npaid	duedate
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	101	101	2	1	1	0	0	0	2023-06-22 23:59:59
2	102	102	3	1	2	100	0	100	2023-07-10 23:59:59
3	103	103	3	2	1	150	150	0	2023-06-08 23:59:59
4	104	104	2	2	0	0	0	0	2024-06-21 23:59:59
5	105	105	4	2	2	300	100	200	2024-03-24 23:59:59
6	106	106	2	2	0	0	0	0	2025-01-31 23:59:59
7	111	111	3	2	1	100	100	0	2024-06-21 23:59:59

Table Name: - Tutor

File

Edit

View

Tools

Help

New Database

Open Database

Write Changes

Revert Changes

Open Project

Save Project

Browse Data

Execute SQL

Database Structure

Edit Pragmas

Table: tutor

Filter in any column

	tutorid	tname	tmid	mobilenno	roomno
	Filter	Filter	Filter	Filter	Filter
1	101	Dr Caroline	caroline@sjctnc.edu.in	9876543210	80
2	102	Dr Ebenezer	ebenezer@sjctnc.edu.in	9856324701	81
3	103	Dr Suresh	suresh@sjctnc.edu.in	9856321470	72
4	104	Dr Benjamin	benjamin@sjctnc.edu.in	9321784561	70
5	105	Dr. Isabella Amali	isabella@sjctnc.edu.in	9988774455	123
6	106	Dr. Justin Marshal	justin@sjctnc.edu.in	7010445566	146

5. SOFTWARE DESCRIPTION

Overview of Python Flask and SQLite

Python Flask

Micro Framework Philosophy:

Flask is a micro framework in Python, it follows the micro framework philosophy, which means it provides the core components needed for web development but leaves other features to extensions. This makes Flask lightweight and flexible, allowing developers to choose and integrate only the components they require for their specific project.

Routing and URL Mapping:

Flask uses a simple and intuitive routing system that maps URLs to Python functions, known as view functions. This mapping is typically done using decorators, making it easy to define routes and their corresponding actions within the application.

For example:

```
from flask import Flask

app = Flask(__name__)

@app.route('/')

def hello_world():

    return 'Hello, World!'
```

Request and Response Handling:

Flask provides request and response objects that encapsulate incoming HTTP requests and outgoing HTTP responses, respectively. This makes it easy to access request data, such as form submissions or URL parameters, and generate appropriate responses.

Flask also supports HTTP methods such as GET, POST, PUT, DELETE, etc., allowing developers to handle different types of requests effectively.

Extensions and Flask Ecosystem:

Flask has a rich ecosystem of extensions that provide additional functionality for tasks such as form validation, authentication, database integration, RESTful API development, and more.

These extensions make it easy to extend Flask's capabilities and integrate with third-party services and libraries.

Development Server and Debugging:

Flask comes with a built-in development server that allows developers to run and test their applications locally during development.

The development server also provides useful debugging features, such as interactive debugger and reloader, which help identify and fix issues quickly.

SQLite

SQLite is a C-language library that implements a small, fast, self-contained, high-reliability, full-featured, SQL database engine. SQLite is the most used database engine in the world. SQLite is built into all mobile phones and most computers and comes bundled inside countless other applications that people use every day. Here we do use DB Browser for SQLite (DB4S) which is a high quality, visual, open source tool to create, design, and edit database files compatible with SQLite.

Embedded Database Engine:

SQLite is a self-contained, serverless, and transactional SQL database engine that is embedded within the application. This means it runs as part of the application process and doesn't require a separate database server to be installed or configured.

SQLite databases are stored in a single disk file, making them easy to manage and distribute along with the application.

Lightweight and Efficient:

SQLite is designed to be lightweight and efficient, making it suitable for embedded systems and applications with low resource requirements.

Despite its small footprint, SQLite provides most of the features expected from a relational database management system, including transactions, indexes, triggers, and views.

Cross-Platform Compatibility:

SQLite is cross-platform and works on various operating systems, including Linux, Windows, macOS, and others. This ensures compatibility and portability of SQLite-based applications across different environments.

SQL Support and Compliance:

SQLite supports a large subset of SQL standard, making it compatible with standard SQL syntax and commonly used database operations.

It provides support for features such as SELECT, INSERT, UPDATE, DELETE statements, as well as complex queries involving joins, subqueries, and aggregates.

Integration of Flask and SQLite

Database Integration with Flask-SQLAlchemy:

Flask can be integrated with SQLite using Flask-SQLAlchemy, which is an extension that provides ORM (Object-Relational Mapping) support for SQLAlchemy, a popular Python SQL toolkit.

SQLAlchemy abstracts database operations using Python objects, allowing developers to interact with the database using high-level Pythonic constructs rather than raw SQL queries.

Database Configuration:

Flask applications can be configured to use SQLite as the backend database by specifying the database URI in the application configuration. The URI typically includes the path to the SQLite database file.

Example :

```
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///casestudy.db'
```

Data Migration and Schema Management:

Flask-SQLAlchemy provides support for database migrations using tools like Alembic, allowing developers to manage changes to the database schema over time.

Migrations enable seamless updates to the database schema without losing existing data or disrupting the application's functionality.

Database Operations:

With Flask-SQLAlchemy, developers can define database models using Python classes that inherit from SQLAlchemy's `db.Model` class. These models represent database tables and their relationships.

Developers can perform various database operations such as querying, inserting, updating, and deleting data using SQLAlchemy's ORM API, which simplifies database interactions and reduces the risk of SQL injection attacks.

In summary, Python Flask and SQLite together provide a powerful yet lightweight solution for building web applications with database functionality. Flask offers a flexible and minimalist web framework, while SQLite provides an embedded and efficient database engine. When combined, they enable developers to create scalable and maintainable web applications with ease.

6. SYSTEM IMPLEMENTATIONS

CODING:

Student side code: //connect.py

```
from flask import*
from flask_sqlalchemy import SQLAlchemy
app=Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///casestudy.db'
db = SQLAlchemy(app)
database={}
class login(db.Model):
    username = db.Column(db.String(80),primary_key=True)
    password = db.Column(db.String(120), unique=True, nullable=False)
    details_f = db.relationship('details', backref='login', lazy=True)
    attendance_f = db.relationship('attendance', backref='login', lazy=True)
    academics_f = db.relationship('academics', backref='login', lazy=True)
    fees_f = db.relationship('fees', backref='login', lazy=True)
    library_f = db.relationship('library', backref='login', lazy=True)
    hostel_f = db.relationship('hostel', backref='login', lazy=True)
class details(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    registerno=db.Column(db.Integer,db.ForeignKey(login.username),unique=True,)
    name=db.Column(db.String(120), unique=False, nullable=False)
    course=db.Column(db.String(50), unique=False, nullable=False)
    sec=db.Column(db.String(5), unique=False, nullable=False)
    cmid=db.Column(db.String(50), unique=True, nullable=False)
    smid=db.Column(db.String(50), unique=True, nullable=False)
    byear=db.Column(db.Integer, unique=False, nullable=False)
    eyear=db.Column(db.Integer, unique=False, nullable=False)
    dob=db.Column(db.String(15), unique=False, nullable=False)
```

```

bloodgroup=db.Column(db.String(10), unique=False, nullable=False)
mobilenno=db.Column(db.Integer, unique=True, nullable=False)
aadharid=db.Column(db.Integer, unique=True, nullable=False)
fathername=db.Column(db.String(50), unique=False, nullable=False)
mothername=db.Column(db.String(50), unique=False, nullable=False)
parentmobilenno=db.Column(db.Integer, unique=True, nullable=False)
resiaddress=db.Column(db.String(60), unique=False, nullable=False)
tutorid=db.Column(db.Integer, unique=False, nullable=False)
stay=db.Column(db.String(15), unique=False, nullable=False)
gender=db.Column(db.String(5), unique=False, nullable=False)
city=db.Column(db.String(30), unique=False, nullable=False)
district=db.Column(db.String(30), unique=False, nullable=False)
pincode=db.Column(db.Integer, unique=False, nullable=False)
#gender,city,district,pincode
class tutor(db.Model):
    tutorid=db.Column(db.String(10),unique=True,primary_key=True)
    tname=db.Column(db.String(120), unique=False, nullable=False)
    tmid=db.Column(db.String(50), unique=True, nullable=False)
    mobilenno=db.Column(db.Integer, unique=True, nullable=False)
    roomno=db.Column(db.Integer, unique=False, nullable=False)
class cclass(db.Model):
    cno=db.Column(db.String(10), primary_key=True, nullable=False)
    course=db.Column(db.String(50), unique=False, nullable=False)
    sec=db.Column(db.String(5), unique=False, nullable=False)
    labno1=db.Column(db.Integer, unique=False, nullable=False)
    labname1=db.Column(db.String(50), unique=False, nullable=False)
    labno2=db.Column(db.Integer, unique=False, nullable=False)
    labname2=db.Column(db.String(50), unique=False, nullable=False)
    labno3=db.Column(db.Integer, unique=False, nullable=False)
    labname3=db.Column(db.String(50), unique=False, nullable=False)
class attendance(db.Model):

```



```

    id = db.Column(db.Integer, primary_key=True)
    registerno=db.Column(db.Integer,db.ForeignKey(login.username),
unique=True)
    course=db.Column(db.String(50), unique=False, nullable=False)
    sub1=db.Column(db.Integer, unique=False, nullable=False)
    sub2=db.Column(db.Integer, unique=False, nullable=False)
    sub3=db.Column(db.Integer, unique=False, nullable=False)
    sub4=db.Column(db.Integer, unique=False, nullable=False)
    sub5=db.Column(db.Integer, unique=False, nullable=False)
    sub6=db.Column(db.Integer, unique=False, nullable=False)
class academics(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    registerno=db.Column(db.Integer,
db.ForeignKey(login.username),unique=True)
    sub1=db.Column(db.String(5), unique=False, nullable=False)
    sub2=db.Column(db.String(5), unique=False, nullable=False)
    sub3=db.Column(db.String(5), unique=False, nullable=False)
    sub4=db.Column(db.String(5), unique=False, nullable=False)
    sub5=db.Column(db.String(5), unique=False, nullable=False)
    sub6=db.Column(db.String(5), unique=False, nullable=False)
    cgpa=db.Column(db.Integer, unique=False, nullable=False)
class fees(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    registerno=db.Column(db.Integer,db.ForeignKey(login.username),
unique=True)
    course=db.Column(db.String(50), unique=False, nullable=False)
    tamt=db.Column(db.Integer, unique=False, nullable=False)
    tpaid=db.Column(db.Integer, unique=False, nullable=False)
    tpending=db.Column(db.Integer, unique=False, nullable=False)
    tduedate=db.Column(db.DateTime, unique=False, nullable=False)
    eamt=db.Column(db.Integer, unique=False, nullable=False)

```

```

    epaid=db.Column(db.Integer, unique=False, nullable=False)
    expending=db.Column(db.Integer, unique=False, nullable=False)
    eduedate=db.Column(db.DateTime, unique=False, nullable=False)
class library(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    registerno=db.Column(db.Integer,
db.ForeignKey(login.username),unique=True )
    borrowedb=db.Column(db.Integer, unique=False, primary_key=False)
    returnedb=db.Column(db.Integer, unique=False, primary_key=False)
    reservedb=db.Column(db.Integer, unique=False, primary_key=False)
    totalf=db.Column(db.Integer, unique=False, primary_key=False)
    paid=db.Column(db.Integer, unique=False, primary_key=False)
    npaid=db.Column(db.Integer, unique=False, primary_key=False)
    duedate=db.Column(db.DateTime, unique=False, nullable=False)
class hostel(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    registerno=db.Column(db.Integer,db.ForeignKey(login.username),
unique=True)
    name=db.Column(db.String(120), unique=False, primary_key=False)
    roomno=db.Column(db.String(5), unique=False, primary_key=False)
    wname=db.Column(db.String(50), unique=False, primary_key=False)
    mfees=db.Column(db.Integer, unique=False, primary_key=False)
    paidstatus=db.Column(db.String(10), unique=False, primary_key=False)
    fine=db.Column(db.Integer, unique=False, primary_key=False)
with app.app_context():
    db.create_all()
    exe=db.session.query(login)
    for i in exe:
        database[i.username]=i.password
@app.route('/')
def initail():

```

```

    return render_template('practice.html')
@app.route('/form_login',methods=['POST','GET'])
def login():
    name1=request.form['Username']
    pwd=request.form['Password']
    if name1 not in database:
        return render_template('practice.html',info='Invalid User')
    else:
        if database[name1]!=pwd:
            return render_template('practice.html',info='Invalid Password')
        else:
            database.clear()
            database[name1]=pwd
            a=db.session.query(details).all()
            return render_template('home.html', ptr=a,k=int(name1))
@app.route('/forget.html')
def forget():
    return render_template('forget.html')
@app.route('/form_forget',methods=['POST','GET'])
def defpass():
    regnum=request.form['registernumber']
    mobnum=request.form['mobilenumber']
    check=db.session.execute(login).filter_by(username==int(regnum))
    print(check)
@app.route('/details.html')
def detail():
    k=0
    value=database.keys()
    for i in value:
        k=int(i)
    a=db.session.query(details).all()

```

```

        return render_template('details.html', ptr=a,k=k)
@app.route('/tutor.html')
def tutorr():
    k=0
    value=database.keys()
    for i in value:
        k=int(i)
    a =
db.session.query(tutor,details).join(details,tutor.tutorid==details.tutorid).all(
)
    for i,j in a:
        print(i.tutorid,j.registerno)
    return render_template('tutor.html',ptr=a,k=k)
@app.route('/class.html')
def cclasss():
    k=0
    value=database.keys()
    for i in value:
        k=int(i)
    a=db.session.query(class,details).join(details,class.sec==details.sec).all()
    for i,j in a:
        print(i.cno,j.name)
    return render_template('class.html',ptr=a,k=k)
@app.route('/attendance.html')
def attendancee():
    k=0
    value=database.keys()
    for i in value:
        k=int(i)
    a=db.session.query(attendance).all()
    return render_template('attendance.html',ptr=a,k=k)

```

```
@app.route('/academics.html')
def academicss():
    k=0
    value=database.keys()
    for i in value:
        k=int(i)
    a=db.session.query(academics).all()
    return render_template('academics.html',ptr=a,k=k)

@app.route('/fees.html')
def feess():
    k=0
    value=database.keys()
    for i in value:
        k=int(i)
    a=db.session.query(fees).all()
    return render_template('fees.html',ptr=a,k=k)

@app.route('/library.html')
def libraryy():
    k=0
    value=database.keys()
    for i in value:
        k=int(i)
    a=db.session.query(library).all()
    return render_template('library.html',ptr=a,k=k)

@app.route('/hostel.html')
def hostell():
    k=0
    value=database.keys()
    for i in value:
        k=int(i)
    a=db.session.query(hostel).all()
```

```

    return render_template('hostel.html',ptr=a,k=k)
@app.route('/practice.html')
def practice():
    return render_template('practice.html')
if __name__ == '__main__':
    app.run(port=5001)

```

Admin Side Coding: //admin.py

```

from flask import Flask, render_template, request, redirect, url_for, session
from flask_sqlalchemy import SQLAlchemy
from datetime import datetime
app = Flask(__name__)
app.secret_key = 'vishalkanuga'
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///casestudy.db'
db = SQLAlchemy(app)
database={}
class Login(db.Model):
    username = db.Column(db.String(80), primary_key=True)
    password = db.Column(db.String(120), nullable=False)
class Details(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    registerno = db.Column(db.Integer, unique=True, nullable=False)
    name = db.Column(db.String(120), nullable=False)
    course = db.Column(db.String(50), nullable=False)
    sec = db.Column(db.String(5), nullable=False)
    cmid = db.Column(db.String(50), unique=True, nullable=False)
    smid = db.Column(db.String(50), unique=True, nullable=False)
    byear = db.Column(db.Integer, nullable=False)
    eyear = db.Column(db.Integer, nullable=False)
    dob = db.Column(db.String(15), nullable=False)

```

```

bloodgroup = db.Column(db.String(10), nullable=False)
mobilenno = db.Column(db.Integer, unique=True, nullable=False)
aadharid = db.Column(db.Integer, unique=True, nullable=False)
fathername = db.Column(db.String(50), nullable=False)
mothername = db.Column(db.String(50), nullable=False)
parentmobilenno = db.Column(db.Integer, unique=True, nullable=False)
resiaddress = db.Column(db.String(60), nullable=False)
tutorid = db.Column(db.Integer, nullable=False)
stay = db.Column(db.String(15), nullable=False)
gender = db.Column(db.String(5), nullable=False)
city = db.Column(db.String(30), nullable=False)
district = db.Column(db.String(30), nullable=False)
pincode = db.Column(db.Integer, nullable=False)
class Tutor(db.Model):
    tutorid = db.Column(db.String(10), primary_key=True)
    tname = db.Column(db.String(120), nullable=False)
    tmid = db.Column(db.String(50), unique=True, nullable=False)
    mobilenno = db.Column(db.Integer, unique=True, nullable=False)
    roomno = db.Column(db.Integer, nullable=False)
class Cclass(db.Model):
    cno = db.Column(db.String(10), primary_key=True)
    course = db.Column(db.String(50), nullable=False)
    sec = db.Column(db.String(5), nullable=False)
    labno1 = db.Column(db.Integer, nullable=False)
    labname1 = db.Column(db.String(50), nullable=False)
    labno2 = db.Column(db.Integer, nullable=False)
    labname2 = db.Column(db.String(50), nullable=False)
    labno3 = db.Column(db.Integer, nullable=False)
    labname3 = db.Column(db.String(50), nullable=False)
class Attendance(db.Model):
    id = db.Column(db.Integer, primary_key=True)

```

```
    registerno = db.Column(db.Integer, nullable=False)
    course = db.Column(db.String(50), nullable=False)
    sub1 = db.Column(db.Integer, nullable=False)
    sub2 = db.Column(db.Integer, nullable=False)
    sub3 = db.Column(db.Integer, nullable=False)
    sub4 = db.Column(db.Integer, nullable=False)
    sub5 = db.Column(db.Integer, nullable=False)
    sub6 = db.Column(db.Integer, nullable=False)
class Academics(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    registerno = db.Column(db.Integer, nullable=False)
    sub1 = db.Column(db.String(5), nullable=False)
    sub2 = db.Column(db.String(5), nullable=False)
    sub3 = db.Column(db.String(5), nullable=False)
    sub4 = db.Column(db.String(5), nullable=False)
    sub5 = db.Column(db.String(5), nullable=False)
    sub6 = db.Column(db.String(5), nullable=False)
    cgpa = db.Column(db.Integer, nullable=False)
class Fees(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    registerno = db.Column(db.Integer, nullable=False)
    course = db.Column(db.String(50), nullable=False)
    tamt = db.Column(db.Integer, nullable=False)
    tpaid = db.Column(db.Integer, nullable=False)
    tpending = db.Column(db.Integer, nullable=False)
    tduedate = db.Column(db.DateTime, nullable=False)
    eamt = db.Column(db.Integer, nullable=False)
    epaid = db.Column(db.Integer, nullable=False)
    epending = db.Column(db.Integer, nullable=False)
    eduedate = db.Column(db.DateTime, nullable=False)
class Library(db.Model):
```



```

id = db.Column(db.Integer, primary_key=True)
registerno = db.Column(db.Integer, nullable=False)
borrowedb = db.Column(db.Integer, nullable=False)
returnedb = db.Column(db.Integer, nullable=False)
reservedb = db.Column(db.Integer, nullable=False)
totalf = db.Column(db.Integer, nullable=False)
paid = db.Column(db.Integer, nullable=False)
npaid = db.Column(db.Integer, nullable=False)
duedate = db.Column(db.DateTime, nullable=False)
class Hostel(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    registerno = db.Column(db.Integer, nullable=False)
    name = db.Column(db.String(120), nullable=False)
    roomno = db.Column(db.String(5), nullable=False)
    wname = db.Column(db.String(50), nullable=False)
    mfees = db.Column(db.Integer, nullable=False)
    paidstatus = db.Column(db.String(10), nullable=False)
    fine = db.Column(db.Integer, nullable=False)
with app.app_context():
    db.create_all()
    exe=db.session.query(Login)
    for i in exe:
        database[i.username]=i.password
# Define routes for the Flask application
@app.route('/')
def index():
    if 'username' in session:
        return render_template('index.html')
    else:
        return redirect(url_for('login'))
@app.route('/add_student', methods=['GET', 'POST'])

```

```
def add_student():
    if request.method == 'POST':
        id=request.form['id']
        registerno = request.form['registerno']
        name = request.form['name']
        course = request.form['course']
        sec=request.form['section']
        cmid=request.form['cmid']
        smid=request.form['smid']
        byear=request.form['byear']
        eyear=request.form['eyear']
        dob=request.form['dob']
        bloodgroup=request.form['bloodgroup']
        mobileno=request.form['mobileno']
        aadharno=request.form['aadharno']
        fathername=request.form['fathername']
        mothername=request.form['mothername']
        parentmobileno=request.form['parentmobileno']
        resiaddress=request.form['resiaddress']
        tutorid=request.form['tutorid']
        stay=request.form['stay']
        gender=request.form['gender']
        city=request.form['city']
        district=request.form['district']
        pincode=request.form['pincode']
        new_student = Details(
            registerno=registerno,
            name=name,
            course=course,
            sec=sec,
            cmid=cmid,
```

```

        smid=smid,
        byear=byear,
        eyear=eyear,
        dob=dob,
        bloodgroup=bloodgroup,
        mobileno=mobileno,
        aadharno=aadharno,
        fathername=fathername,
        mothername=mothername,
        parentmobileno=parentmobileno,
        resiaddress=resiaddress,
        tutorid=tutorid,
        stay=stay,
        gender=gender,
        city=city,
        district=district,
        pincode=pincode
    )
    db.session.add(new_student)
    db.session.commit()

    return redirect(url_for('index'))
else:
    return render_template('add_student.html')
@app.route('/add_tutor', methods=['GET', 'POST'])
def add_tutor():
    if request.method == 'POST':
        tutorid = request.form['tutorid']
        tname = request.form['tname']
        tmid=request.form['tmid']
        mobileno=request.form['mobileno']

```

```

roomno=request.form['roomno']
new_tutor = Tutor(
    tutorid=tutorid,
    tname=tname,
    tmid=tmid,
    mobileno=mobileno,
    roomno=roomno
)
db.session.add(new_tutor)
db.session.commit()
return redirect(url_for('index'))
else:
    return render_template('add_tutor.html')
@app.route('/add_academics', methods=['GET', 'POST'])
def add_academics():
    if request.method == 'POST':
        id = request.form['id']
        registerno = request.form['registerno']
        sub1 = float(request.form['sub1'])
        sub2 = float(request.form['sub2'])
        sub3 = float(request.form['sub3'])
        sub4 = float(request.form['sub4'])
        sub5 = float(request.form['sub5'])
        sub6 = float(request.form['sub6'])
        cgpa = float(request.form['cgpa'])
        new_academics = Academics(
            id=id,
            registerno=registerno,
            sub1=sub1,
            sub2=sub2,
            sub3=sub3,

```

```

        sub4=sub4,
        sub5=sub5,
        sub6=sub6,
        cgpa=cgpa
    )
    db.session.add(new_academics)
    db.session.commit()
    return redirect(url_for('index'))
else:
    return render_template('add_academics.html')
@app.route('/add_attendance', methods=['GET', 'POST'])
def add_attendance():
    if request.method == 'POST':
        id = request.form['id']
        registerno = request.form['registerno']
        course=request.form['course']
        sub1=request.form['sub1']
        sub2=request.form['sub2']
        sub3=request.form['sub3']
        sub4=request.form['sub4']
        sub5=request.form['sub5']
        sub6=request.form['sub6']
        new_attendance = Attendance(
            id=id,
            registerno=registerno,
            course=course,
            sub1=sub1,
            sub2=sub2,
            sub3=sub3,
            sub4=sub4,
            sub5=sub5,

```

```

        sub6=sub6
    )
    db.session.add(new_attendance)
    db.session.commit()
    return redirect(url_for('index'))
else:
    return render_template('add_attendance.html')
@app.route('/add_class', methods=['GET', 'POST'])
def add_class():
    if request.method == 'POST':
        cno = request.form['cno']
        course=request.form['course']
        sec=request.form['sec']
        labno1=request.form['labno1']
        labname1=request.form['labname1']
        labno2=request.form['labno2']
        labname2=request.form['labname2']
        labno3=request.form['labno3']
        labname3=request.form['labname3']
        new_class = Cclass(
            cno=cno,
            course=course,
            sec=sec,
            labno1=labno1,
            labname1=labname1,
            labno2=labno2,
            labname2=labname2,
            labno3=labno3,
            labname3=labname3
        )
    db.session.add(new_class)

```

```

        db.session.commit()

        return redirect(url_for('index'))

    else:

        return render_template('add_class.html')

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

    if request.method == 'POST':

        id = request.form['id']

        registerno=request.form['registerno']

        course=request.form['course']

        tamt=request.form['tamt']

        tpaid=request.form['tpaid']

        tpending=request.form['tpending']

        eamt=request.form['eamt']

        epaid=request.form['epaid']

        epending=request.form['epending']

        tduedate_date = request.form['tduedate_date']

        tduedate_time = request.form['tduedate_time']

        tduedate = datetime.strptime(tduedate_date + ' ' + tduedate_time, '%Y-%m-%d %H:%M:%S')

        eduedate_date = request.form['eduedate_date']

        eduedate_time = request.form['eduedate_time']

        eduedate = datetime.strptime(eduedate_date + ' ' + eduedate_time, '%Y-%m-%d %H:%M:%S')

        new_fees = Fees(

            id=id,

            registerno=registerno,

            course=course,

            tamt=tamt,

            tpaid=tpaid,

            tpending=tpending,

```

```

        tduedate=tduedate,
        eamt=eamt,
        epaid=epaid,
        epending=epending,
        eduedate=eduedate
    )
    db.session.add(new_fees)
    db.session.commit()
    return redirect(url_for('index'))
else:
    return render_template('add_fees.html')
@app.route('/add_hostel', methods=['GET', 'POST'])
def add_hostel():
    if request.method == 'POST':
        id = request.form['id']
        registerno=request.form['registerno']
        name=request.form['name']
        roomno=request.form['roomno']
        wname=request.form['wname']
        mfees=request.form['mfees']
        paidstatus=request.form['paidstatus']
        fine=request.form['fine']
        new_hostel = Hostel(
            id=id,
            registerno=registerno,
            name=name,
            roomno=roomno,
            wname=wname,
            mfees=mfees,
            paidstatus=paidstatus,
            fine=fine

```



```

    )
    db.session.add(new_hostel)
    db.session.commit()
    return redirect(url_for('index'))
else:
    return render_template('add_hostel.html')
@app.route('/add_library', methods=['GET', 'POST'])
def add_library():
    if request.method == 'POST':
        id = request.form['id']
        registerno=request.form['registerno']
        borrowedb=request.form['borrowedb']
        returnedb=request.form['returnedb']
        reservedb=request.form['reservedb']
        totalf=request.form['totalf']
        paid=request.form['paid']
        npaid=request.form['npaid']
        duedate_date = request.form['duedate_date']
        duedate_time = request.form['duedate_time']
        duedate = datetime.strptime(duedate_date + ' ' + duedate_time, '%Y-%m-%d %H:%M:%S')
        new_library = Library(
            id=id,
            registerno=registerno,
            borrowedb=borrowedb,
            returnedb=returnedb,
            reservedb=reservedb,
            totalf=totalf,
            paid=paid,
            npaid=npaid,
            duedate=duedate

```

```

    )
    db.session.add(new_library)
    db.session.commit()
    return redirect(url_for('index'))
else:
    return render_template('add_library.html')
@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']
        user = Login.query.filter_by(username=username,
password=password).first()
        if user:
            # Storing the username in the session
            session['username'] = user.username
            return redirect(url_for('index'))
        else:
            return render_template('login.html', error='Invalid username or
password.')
        else:
            return render_template('login.html')
if __name__ == '__main__':
    app.run(port=5000)

```

/ sample html template code : practice.html */*

<!DOCTYPE html>

<head>

<meta charset="UTF-8">

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
<title>Students Login</title>
```

```
<style>
```

```
  body {
```

```
    margin: 0;
```

```
    padding: 5;
```

```
    background-color: white;
```

```
    font-family: Arial, sans-serif;
```

```
    background-image: url("/static/white1.jpg");
```

```
    background-size: cover;
```

```
    background-position: center top;
```

```
    background-repeat: no-repeat;
```

```
  }
```

```
  .container {
```

```
    flex: 1;
```

```
    position: absolute;
```

```
    top: 50%;
```

```
    left: 50%;
```

```
    transform: translate(-50%, -50%);
```

```
    background-color: rgb(0 0 0 / 50%);
```

```
    padding: 20px;
```

```
border-radius: 8px;  
  
box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);  
  
width: 300px;  
  
text-align: center;  
  
}  
  
.formgroup {  
  
    margin-bottom: 20px;  
  
}  
  
.formgroup label {  
  
    display: block;  
  
    margin-bottom: 8px;  
  
    font-weight: bold;  
  
    color: white;  
  
}  
  
.formgroup input {  
  
    width: 100%;  
  
    padding: 8px;  
  
    margin-bottom: 16px;  
  
    box-sizing: border-box;  
  
    border: 1px solid #ccc;  
  
    border-radius: 4px;
```

```
}
```

```
button {
```

```
    padding: 10px;
```

```
    margin: 10px 0;
```

```
    border: none;
```

```
    cursor: pointer;
```

```
    background-color: green;
```

```
    color: white;
```

```
    border-radius: 4px;
```

```
}
```

```
button:hover {
```

```
    background-color: darkgreen;
```

```
}
```

```
.extra {
```

```
    display: flex;
```

```
    align-items: center;
```

```
    justify-content: space-between;
```

```
}
```

```
a {
```

```
    text-decoration: none;
```

```
    color: green;
```

```
}
```

```
a:hover {
```

```
    text-decoration: underline;
```

```
}
```

```
footer {
```

```
    position: absolute;
```

```
    bottom: 0;
```

```
    right: 0;
```

```
    padding: 10px;
```

```
    text-align: center;
```

```
    color: black;
```

```
}
```

```
h1 {
```

```
    color: white;
```

```
}
```

```
.blurred-logo {
```

```
    filter: blur(5px);
```

```
}
```

```
.forget-password {
```

```
    color: white;
```

```
}
```

</style>

</head>

<body style="background-color:white">

<p style="background-color: rgb(18, 18, 57); margin: 0;"></p>

<p></p>

<form action="/form_login" method="post">

<div class="container">

<div class="formgroup">

<label>Username</label>

<input style="background-color: white;" type="text" placeholder="Enter Username " name="Username" required>

<label>Password</label>

<input style="background-color: white;" type="password" placeholder="Enter Password" name="Password" required>

</div>

<div class="extra">

<button type="submit" style="background-color: green; color: white; border-color: whitesmoke;">Login</button>

Forget Password?

</div>

<h1>{{info}}</h1>

</div>

</form>

<footer>

© 2024 Sjctnc. All rights reserved.

</footer>

</body>

7. TESTING

Testing is a process of executing a program with the interest of finding an error. A good test is one that has high probability of finding the yet undiscovered error. Testing should systematically uncover different classes of errors in a minimum amount of time with a minimum amount of efforts. Two classes of inputs are provided to test the process

- A software configuration that includes a software requirement specification, a design specification and source code.
- A software configuration that includes a test plan and procedure, any testing tool and test cases and their expected results.

Testing is divided into several distinct operations:

1. Unit Testing

Unit test comprises of a set tests performed by an individual program prior to the integration of the unit into large system. A program unit is usually the smallest free functioning part of the whole system. Module unit testing should be as exhaustive as possible to ensure that each representation handled by each module has been tested. All the units that makeup the system must be tested independently to ensure that they work as required.

During unit testing some errors were raised and all of them were rectified and handled well. The result was quiet satisfactory and it worked well.

2. Integration Testing

Integration testing is a system technique for constructing the program structure while at the same time conducting tests to uncover errors associated with interfacing. The objective is to take unit tested modules and build a program structure that has been dictated by design. Bottom-up integration is the traditional strategy used to integrate the components of a software system into functioning whole

The system was done the integration testing. All the modules were tested for their compatibility with other modules .They test was almost successful.

3. Validation Testing

After validation testing, software is completely assembled as a package, interfacing errors that have been uncovered and corrected and the final series of

software test; the validation test begins. Steps taken during software design and testing can greatly improve the probability of successful integration in the larger system. System testing is actually a series of different tests whose primary purpose is to fully exercise the compute –based system.

4. Recovery Testing

It is a system that forces the software to fail in a variety of ways and verifies that the recovery is properly performed.

5. Security Testing

It attempts to verify that protection mechanisms built into a system will in fact protect it from improper penetration. The system's security must of course be tested from in vulnerability form frontal attack.

6. Stress Testing

Stress tools are designed to confront programs with abnormal situations. Stress testing executes a system in a manner that demands resources in abnormal quantity and volume.

7. Black Box Testing

Black box testing is done to find out the following information as shown in below:

- Incorrect or missing functions.
- Interface errors.
- Errors or database access.
- Performance error.
- Termination error.

The mentioned testing is carried out successfully for this application according to the user's requirement specification.

8. Test Data Output

After preparing test data, the system under study is tested using the test data. While testing the system using test data, errors are again uncovered and corrected by using above testing and corrections are also noted for future use.

8. APPENDIX

Sample Screen Shots

Student Login:

The image shows a web browser window with the address bar displaying '127.0.0.1:5001'. The page title is 'ST. JOSEPH'S COLLEGE OF ARTS AND SCIENCE (AUTONOMOUS)', affiliated to Annamalai University, with NAAC 3rd Cycle Re-Accreditation with 'A' Grade, Cuddalore-607001. The main content area features a login form with a 'Username' label, an 'Enter Username' input field, a 'Password' label, an 'Enter Password' input field, a green 'Login' button, and a 'Forgot Password?' link. The background has a faint, repeating diamond pattern.

Student Details:



ST. JOSEPH'S COLLEGE OF ARTS AND SCIENCE (AUTONOMOUS),
Affiliated to Annamalai University,
NAAC 3rd Cycle Re-Accreditation with "A" Grade,
Cuddalore-607001

127.0.0.1:5001/form_login

Details

Tutor

Class

attendance

Academic

Fees

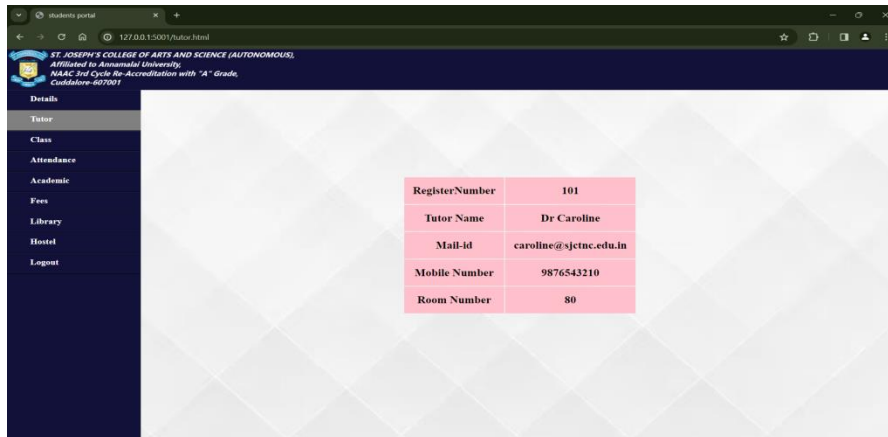
Library

Hostel

Logout

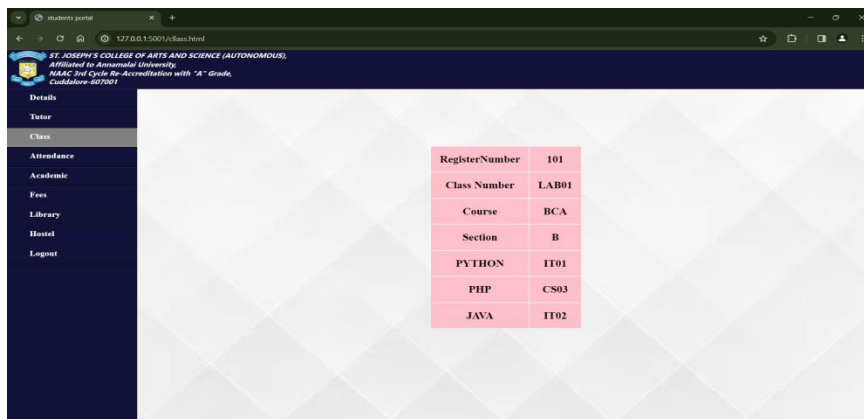
RegisterNumber	101
Name	Vishal
Course	BCA
Year Of Study	2021-2024
College Mail-id	s101@gmail.com
Personal Mail-id	vishal@gmail.com
Date Of Birth	14-07-2004
Mobile Number	9876543210
City	Panruti

Tutor details:



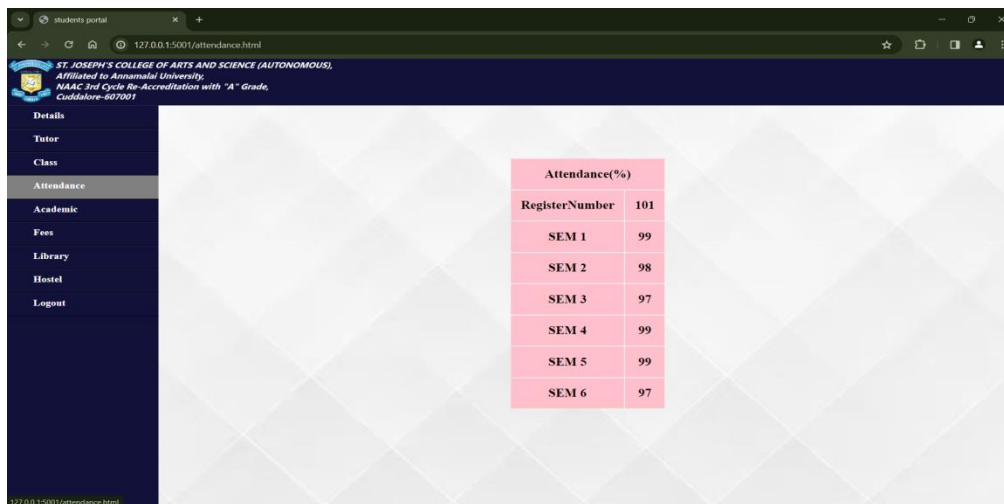
RegisterNumber	101
Tutor Name	Dr Caroline
Mail-id	caroline@sjctnc.edu.in
Mobile Number	9876543210
Room Number	80

Class details:



RegisterNumber	101
Class Number	LAB01
Course	BCA
Section	B
PYTHON	IT01
PHP	CS03
JAVA	IT02

Attendance details:



Attendance(%)	
RegisterNumber	101
SEM 1	99
SEM 2	98
SEM 3	97
SEM 4	99
SEM 5	99
SEM 6	97

Academic details:



The screenshot shows the 'Academic' section of the student portal. The left sidebar contains links for Details, Tutor, Class, Attendance, Academic, Fees, Library, Hostel, and Logout. The main content area displays a table titled 'MARKS(GPA)' for RegisterNumber 101, showing GPA values for SEM 1 through SEM 6.

MARKS(GPA)	
RegisterNumber	101
SEM 1	6.2
SEM 2	6.8
SEM 3	6.4
SEM 4	6.0
SEM 5	5.0
SEM 6	7.2

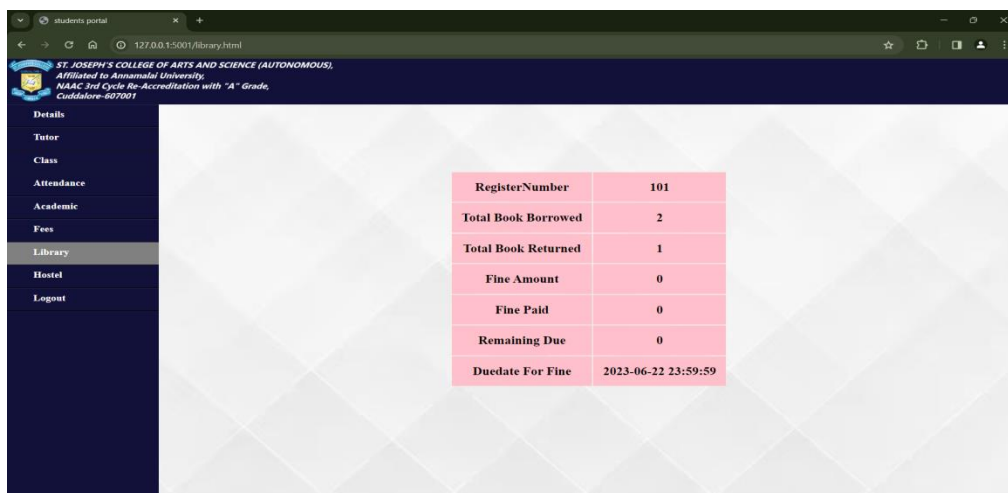
Fees details:



The screenshot shows the 'Fees' section of the student portal. The left sidebar contains links for Details, Tutor, Class, Attendance, Academic, Fees, Library, Hostel, and Logout. The main content area displays a table with fee details for RegisterNumber 101, including Course, Total Amount, Total Paid, Pending, Due Date, Exam Fees, Exam Fees Paid, and Due Date.

RegisterNumber	101
Course	BCA
Total Amount	50000
Total Paid	50000
Pending	0
Due Date	2023-02-18 23:59:59
Exam Fees	2000
Exam Fees Paid	2000
Pending	0
Due Date	2023-06-22 23:59:59

Library details:



The screenshot shows the 'Library' section of the student portal. The left sidebar contains links for Details, Tutor, Class, Attendance, Academic, Fees, Library, Hostel, and Logout. The main content area displays a table with library details for RegisterNumber 101, including Total Book Borrowed, Total Book Returned, Fine Amount, Fine Paid, Remaining Due, and Due Date For Fine.

RegisterNumber	101
Total Book Borrowed	2
Total Book Returned	1
Fine Amount	0
Fine Paid	0
Remaining Due	0
Due Date For Fine	2023-06-22 23:59:59

Hostel details:

The screenshot shows a web browser window with the address bar displaying '127.0.0.1:5001/hostel.html'. The page header includes the college logo and name: 'ST. JOSEPH'S COLLEGE OF ARTS AND SCIENCE (AUTONOMOUS), Affiliated to Annamalai University, NAAC 3rd Cycle Re-Accredited with "A" Grade, Cuddalore-607001'. A left sidebar contains a menu with items: Details, Tutor, Class, Attendance, Academic, Fees, Library, Hostel (highlighted), and Logout. The main content area features a table with the following data:

RegisterNumber	101
Name	Vishal
Room Number	100
Wardern Name	DAVID
Mess Fees	2050
Paid Status	YES
Fine	0

Admin panel:

The screenshot shows a web browser window with the address bar displaying '127.0.0.1:5000/login'. The page header is a dark bar with the text 'Welcome to the Admin Panel'. The main content area has a light gray diamond-patterned background. In the center, there is a white login box with the title 'Login'. Inside the box, there are two input fields labeled 'Username' and 'Password', and a green 'Login' button at the bottom.

Index page:

The screenshot shows a web browser window with the address bar displaying '127.0.0.1:5000'. The page header is a dark bar with the text 'Welcome to the Admin Panel'. The main content area has a light gray diamond-patterned background. In the center, there is a white box containing a vertical list of buttons: 'Add Student', 'Add Tutor', 'Add Academics', 'Add Attendance', 'Add Class Details', 'Add Fee Details', 'Add Hostel Details', 'Add Library Details', and 'Add Login Details'. At the bottom of the page, there is a dark footer bar with the text '© 2024 Admin Panel'.

9. CONCLUSION & FUTURE ENHANCEMENT

The project Students Portal is completed, satisfying the required design specifications. The system provides a user-friendly interface. The software is developed with modular approach. All modules in the system have been tested with valid data and invalid data and everything work successfully. Thus the system has fulfilled all the objectives identified and is able to replace the existing system. The constraints are met and overcome successfully. The system is designed as like it was decided in the design phase. It can be enhanced further in future in following ways if needed:

By Centralized Database:

To make the common database servers with the help of Firebase, aws or any other Servers to maintain all the records/data .

Online Payment method:

To make a payment for the hostel fee, mess, tuition and exam fees through online by connecting the online payment platform like UPI, G-Pay and so.

Dynamic Pages:

To add more features like updating, deleting the details from admin side so that the changes can be made more easily and dynamically.

Data Verification:

To add pages in student side for feedback and changes and much more.

The application has been tested with live data and has provided a successful result. Hence the software has proved to work efficiently.

10. REFERNCES

BOOKS:

- Core Python Programming - Python for programming the Software.
- W3Schools- HTML and CSS For Front End

ONLINE REFERENCE:

www.openai.com

www.google.com

www.github.com

www.geekforgeeks.com