College Due Management System

The College Due Management System is a comprehensive web-based application designed to efficiently store and manage the due details of students for future reference during semester registration. The system provides a centralized platform that aids teachers, students and staff in monitoring their dues across various sectors.

In traditional college environments, keeping track of dues can be a cumbersome and time-consuming task. This project addresses this challenge by offering an intuitive and user-friendly interface that streamlines the entire process. Students can log in to the system to view their outstanding dues.

The system empowers teachers and staff by enabling them to record and update student's dues easily. They can access a comprehensive overview of their student's financial status, making it convenient for them to guide and support students in fulfilling their obligations promptly.

The College Due Management System utilizes a secure database to safeguard sensitive financial information, ensuring the privacy and confidentiality of student records. Administrators can manage user roles and access permissions to maintain data integrity and protect against unauthorized access.

Moreover, the system is highly scalable, allowing for seamless integration with existing college management systems. It can be customized to accommodate specific college requirements and expanded to include additional features such as generating reports, sending automated reminders, and integrating with online payment gateways.

Through the implementation of the College Due Management System, colleges can enhance efficiency, transparency, and accountability in managing dues. Students can stay informed about their financial responsibilities, and teachers can proactively assist them in resolving any outstanding issues. Overall, this project contributes to a more organized and productive college environment, benefitting both students and faculty members alike.

CONTENTS

LIST OF FIGURES

ABBREVIATIONS

☐	UML	- Unified Modelling Language
☐	UI	- User Interface
☐	DFD	- Data Flow Diagram
☐	GUI	- Graphical User Interface

CHAPTER 1: INTRODUCTION

In the dynamic realm of modern education, managing student dues efficiently has become a crucial aspect for educational institutions. With the increasing number of courses, programs, and activities offered, keeping track of student's outstanding dues has become a challenging task. To address this issue and ensure a seamless academic experience, we present the "College Due Management System" – a robust and user-friendly software solution designed to simplify and streamline the management of student dues.

The College Due Management System is a comprehensive digital platform that facilitates the storage and retrieval of student's due details for future reference during semester registration. This innovative system empowers both teachers and students to effortlessly monitor their dues across various academic and non-academic sectors within the college.

1.1 Key Features:

1. Centralized Database: The system maintains a centralized database to store all relevant information related to student's dues. This ensures easy access and retrieval of data, reducing paperwork and enhancing overall efficiency.

2. **Real-Time Updates:** The College Due Management System provides real-time updates, enabling students and faculty members to view their current due status. This feature ensures prompt action and prevents any potential delays in the registration process.

3. User-Friendly Interface: The user-friendly interface of the system ensures that both students, teachers and staff can navigate and interact with the platform effortlessly. Minimal training is required to use the system effectively.

4. Secure Authentication: To protect sensitive student information, the College Due Management System employs robust security measures like secure authentication protocols.

5. Scalable Architecture: Built with scalability in mind, the system can accommodate the growing needs of the college as the number of students and courses increase.

The College Due Management System is not only a tool for record-keeping but also a strategic asset that enhances administrative efficiency and student satisfaction. By implementing this system, the college aims to foster a conducive learning environment where students can focus on their academics without worrying about dues and administrative hurdles.

In the following sections of this project report, we will delve into the system's design, development, implementation, and the benefits it brings to the college community. We will also explore the challenges faced during the development process and the lessons learned, along with future recommendations for further improvement.

With the College Due Management System, the college takes a significant step towards embracing technology to streamline administrative processes and elevate the overall academic experience for its students and faculty alike.

1.2 Objective

The objective of the product is that it would eradicate the paperwork done nowadays. And reduce the time consumption needed to sort those documents. This would help to view the due details in one platform instead of the burden of going through each of it individually.

1.3 Scope

The present system for due management requires the student to be physically present at each sector for collecting their no due certificate. So the digitalization of this process saves the student and the staff their time and effort.

Thus using this product can make it more comfortable and effective for the future generation too.

CHAPTER 2: SOFTWARE REQUIREMENT SPECIFICATION

This chapter will discuss product overview, product functionality, design and implementation constraints. It also discusses specific system requirements.

2.1 Product Overview

The College Due Management System is a cutting-edge software solution designed to efficiently store and manage student dues for future reference during semester registration. It empowers teachers, students and staff with real-time access to outstanding dues across various academic and non-academic sectors within the college. The system offers a centralized database ensuring streamlined administrative processes and enhanced student satisfaction. With its user-friendly interface and secure authentication, the College Due Management System is a strategic asset that simplifies record-keeping, fosters a conducive learning environment, and allows educational institutions to focus on delivering a seamless academic experience.

2.2 Product Functionality

- Reduces the hasty amount of paperwork
- Streamlined due collection process.
- Upgraded communication between members.
- Enhanced student experience.
- Single center point for the process.
- Reduces Human Error

2.3 Design and Implementation Constraints

- The time limit for the completion of this project is till the end of this semester.
- The possibility of the server being down during production due to some internal errors.
- The server should be able to handle heavy traffic.
- The database should be updated in real-time.
- Should be able to handle the increasing number of students.
- The UI should be easy to use.

2.4 Requirements

2.4.1 Hardware Requirements

- A system with processor Intel i3 or AMD Ryzen 3 or above
- RAM capacity 4GB or above

2.4.2 Software Requirements

- Web browser (preferably Google Chrome)
- Programming Language - Python
- Packages: Flask, Jinja2, mysql-connector-python
- Database - MySQL

2.5 Functional Requirements

- Students should be able to view their individual dues
- Tutors should be able to view the dues of all the students of his/her class
- Staff should be able to view and edit the dues of their respective sector
- Admin should be able to create, edit and delete users

CHAPTER 3: SYSTEM ARCHITECTURE AND DESIGN

Design process is the process through which designers design interfaces in software systems with an emphasis on aesthetics or style is termed as UI design. Here we use different design processes like data flow diagram, activity diagram, use case diagram to implement our project.

3.1 Data Flow Diagram

Data Flow diagram (DFD) maps out the flow of information for any process or system. It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination.

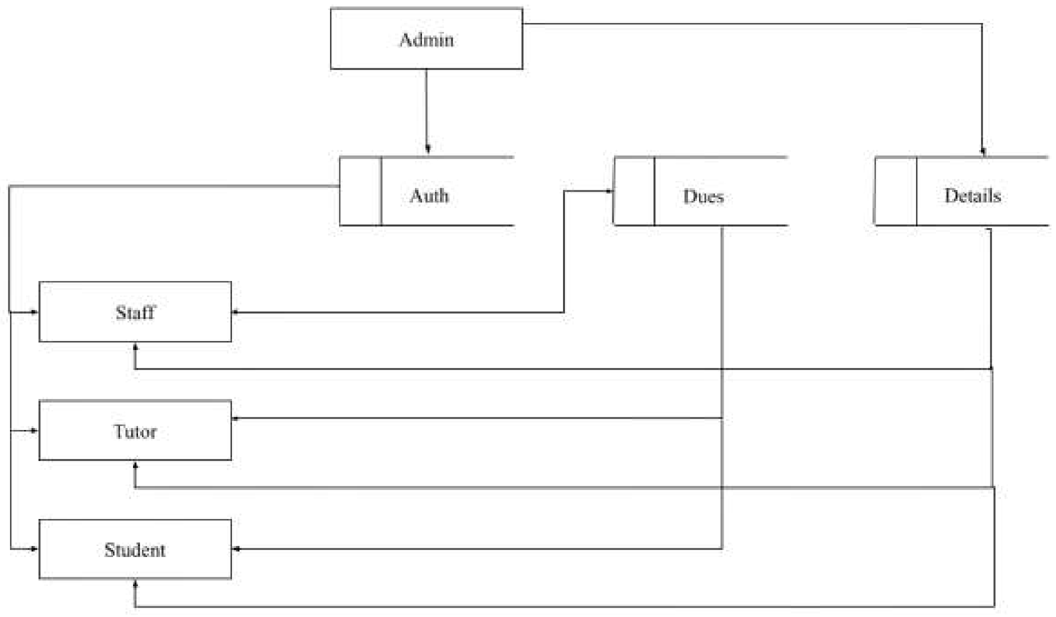


Fig 3.1 Data Flow Diagram

3.2 Use Case Diagram

In the Unified Modelling Language (UML), a use case diagram can summarize the details of your system's users (also known as actors) and their interaction with the system. To build one, you'll use a set of specialized symbols and connectors. An effective use case diagram can help your team discuss and represent:

- Scenarios in which your system or application interact with people, organizations, or external systems
- Goals that your system or application helps those entities (known as actors) achieve.
- The scope of your system.

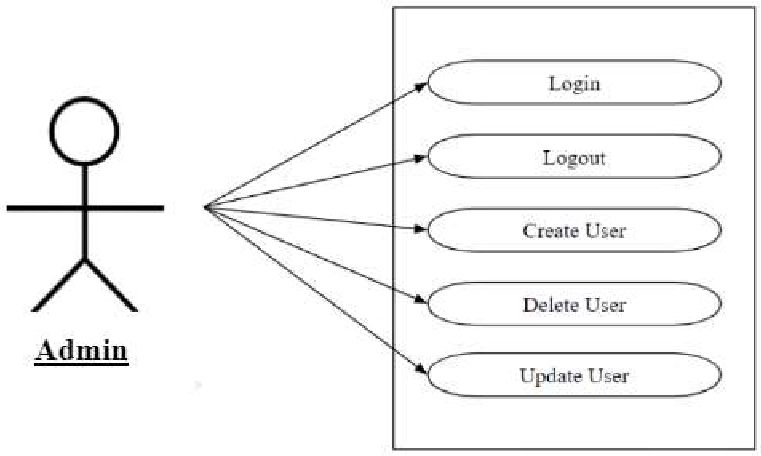


Fig 3.2 Admin Use-case Diagram

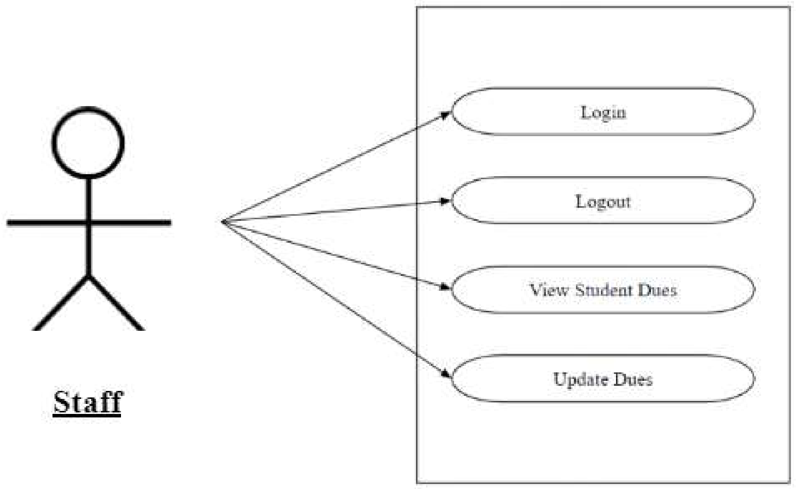


Fig 3.3 Staff Use-case Diagram

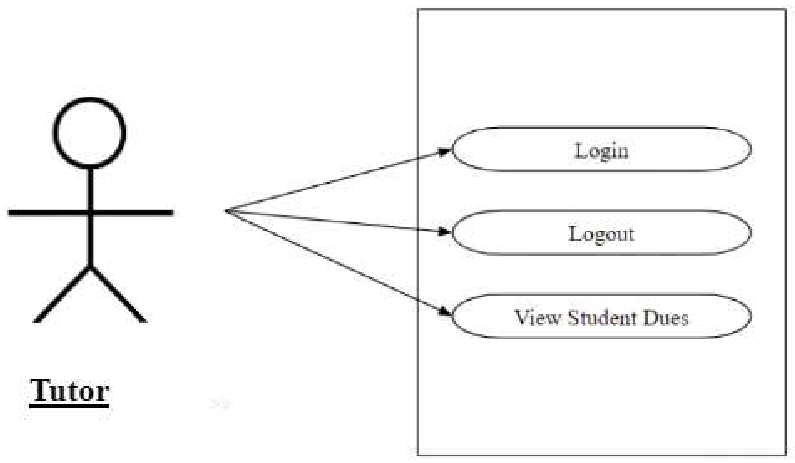


Fig 3.4 Tutor Use-case Diagram

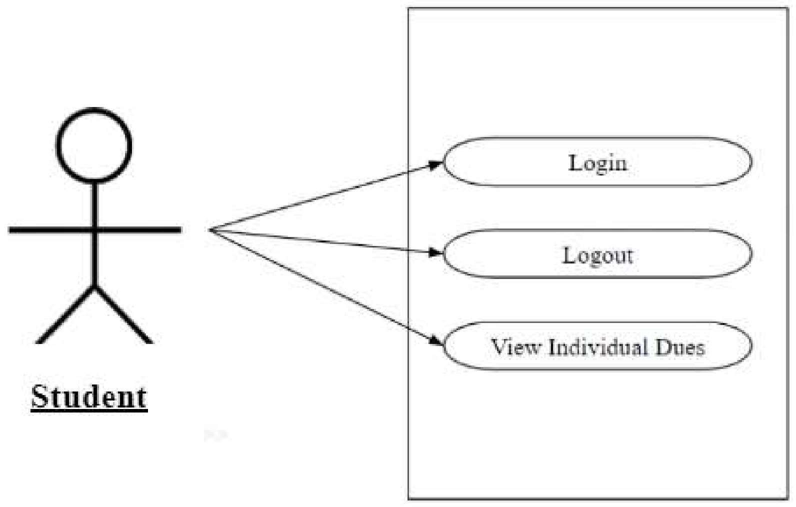


Fig 3.5 Student Use-case Diagram

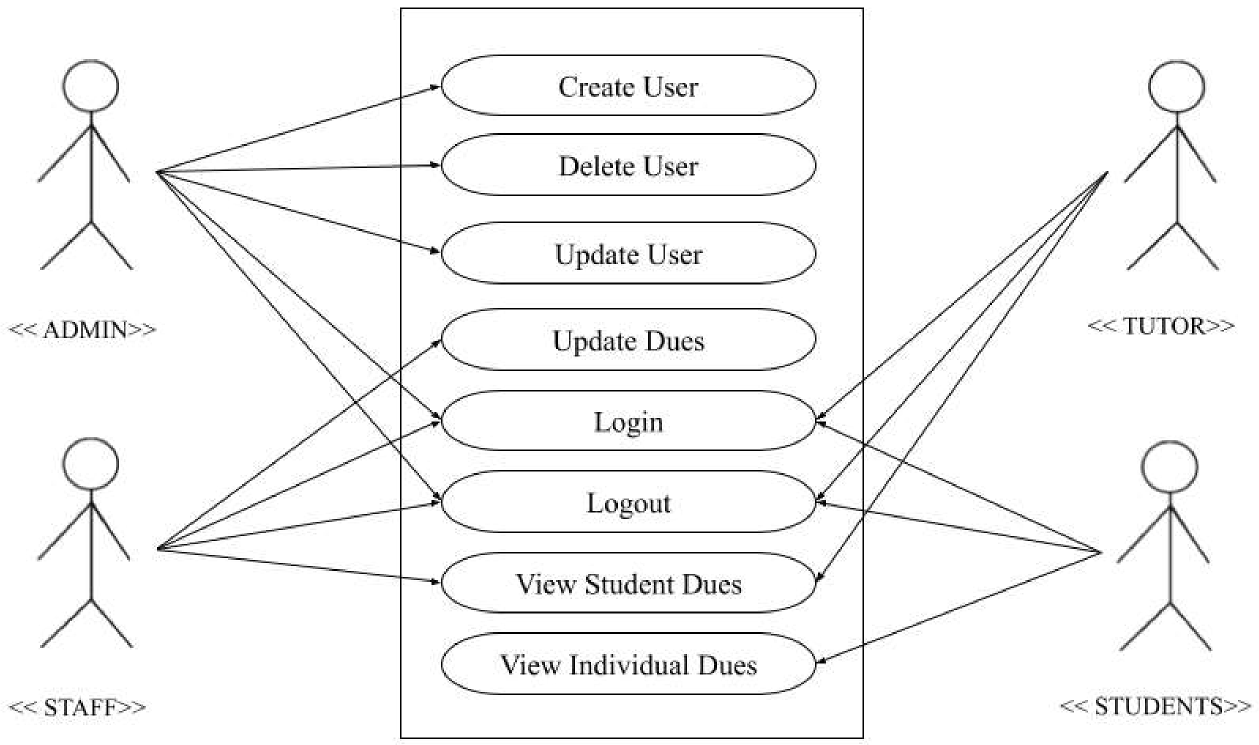


Fig 3.6 Use-case Diagram

CHAPTER 4: IMPLEMENTATION AND TESTING

In the implementation phase of the College Due Management System project, a careful selection of tools and technologies played a pivotal role in ensuring the system's robustness, efficiency, and scalability. This chapter outlines the key tools and technologies employed during the development process, highlighting their significance in achieving project goals.

4.1 Programming Languages and Templating

The system's backend was developed using Python, a versatile and widely-used programming language known for its simplicity and extensive libraries. To seamlessly integrate dynamic content into web pages, Jinja2, a powerful templating engine, was employed. Flask, a lightweight and flexible web framework, was utilized to build the application's backend. It enabled seamless routing, data processing, and interaction with the database, contributing to the system's overall robustness and responsiveness.

4.1.1 Flask

Flask is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries.It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools

Fig 4.1 Flask

4.1.2 Jinja

Jinja is a web template engine for the Python programming language. It was created by Armin Ronacher and is licensed under a BSD License. Jinja is similar to the Django template engine but provides Python-like expressions while ensuring that the templates are evaluated in a sandbox. It is a text-based template language and thus can be used to generate any markup as well as source code.

Fig 4.2 Jinja

The Jinja template engine allows customization of tags, filters, tests, and globals. Also, unlike the Django template engine, Jinja allows the template designer to call functions with arguments on objects. Jinja is Flask's default template engine and it is also used by Ansible, Trac, and Salt.

4.2 Frontend Technologies

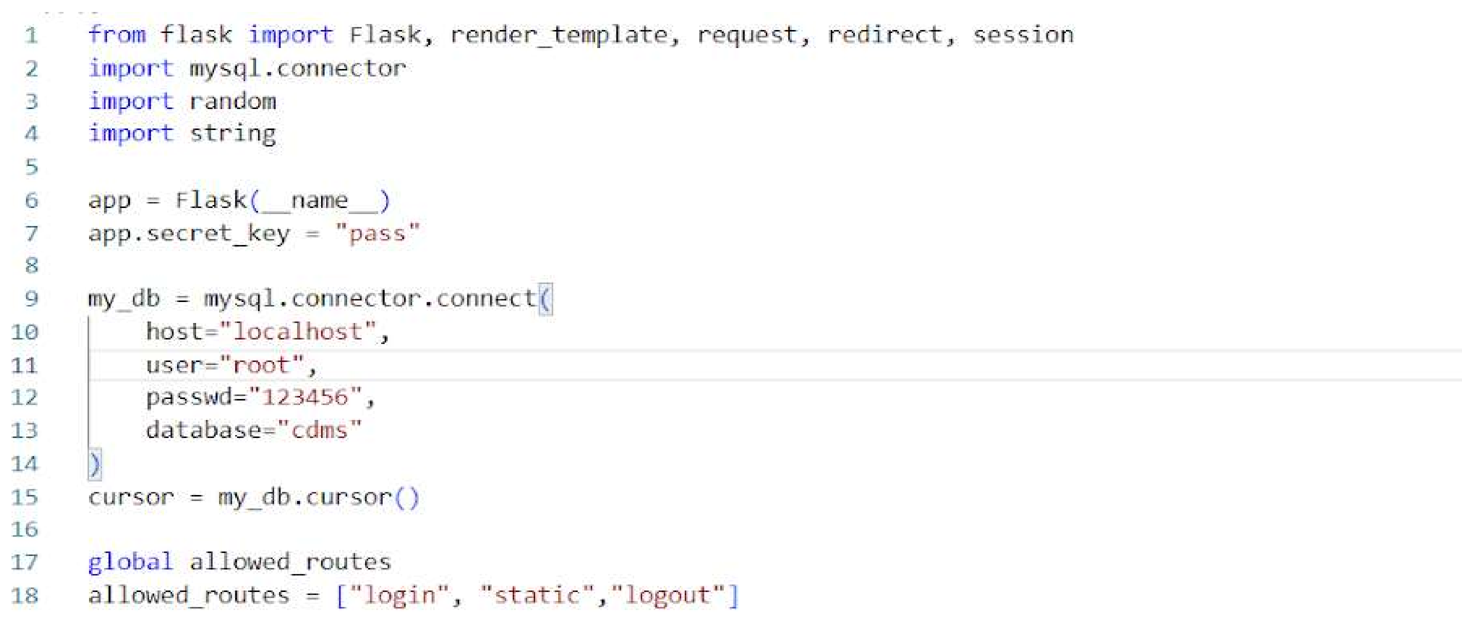
For the user interface, a combination of HTML5, CSS3, and JavaScript was employed. HTML5 provided the structure of the web pages, while CSS3 handled the styling, ensuring a visually appealing and responsive design. JavaScript facilitated dynamic user interactions like searching the table, enhancing the overall user experience.

4.3 Database Management

MySQL, a robust relational database management system, was strategically chosen to effectively store and manage the system's data. Its proven scalability, data integrity features, and compatibility with various programming languages made it an ideal choice for maintaining student due records securely and efficiently. To establish seamless connectivity between the Python backend and the MySQL database, the mysql-connector-python library was employed. This library facilitated the execution of SQL queries, insertion, and retrieval of data from the database, ensuring reliable and optimized data management within the College Due Management System.

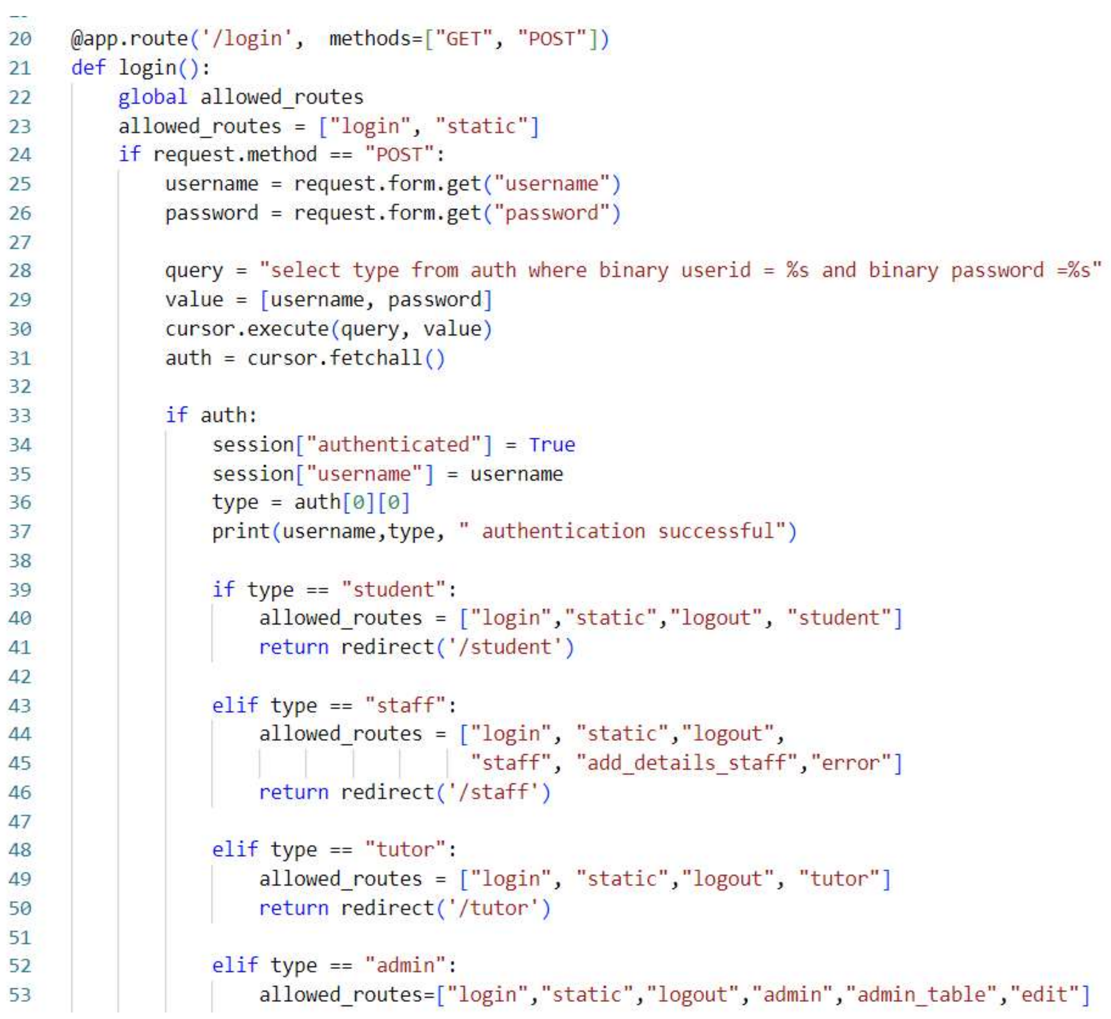
4.4 Sample Code

Importing necessary modules and initializing database credentials.



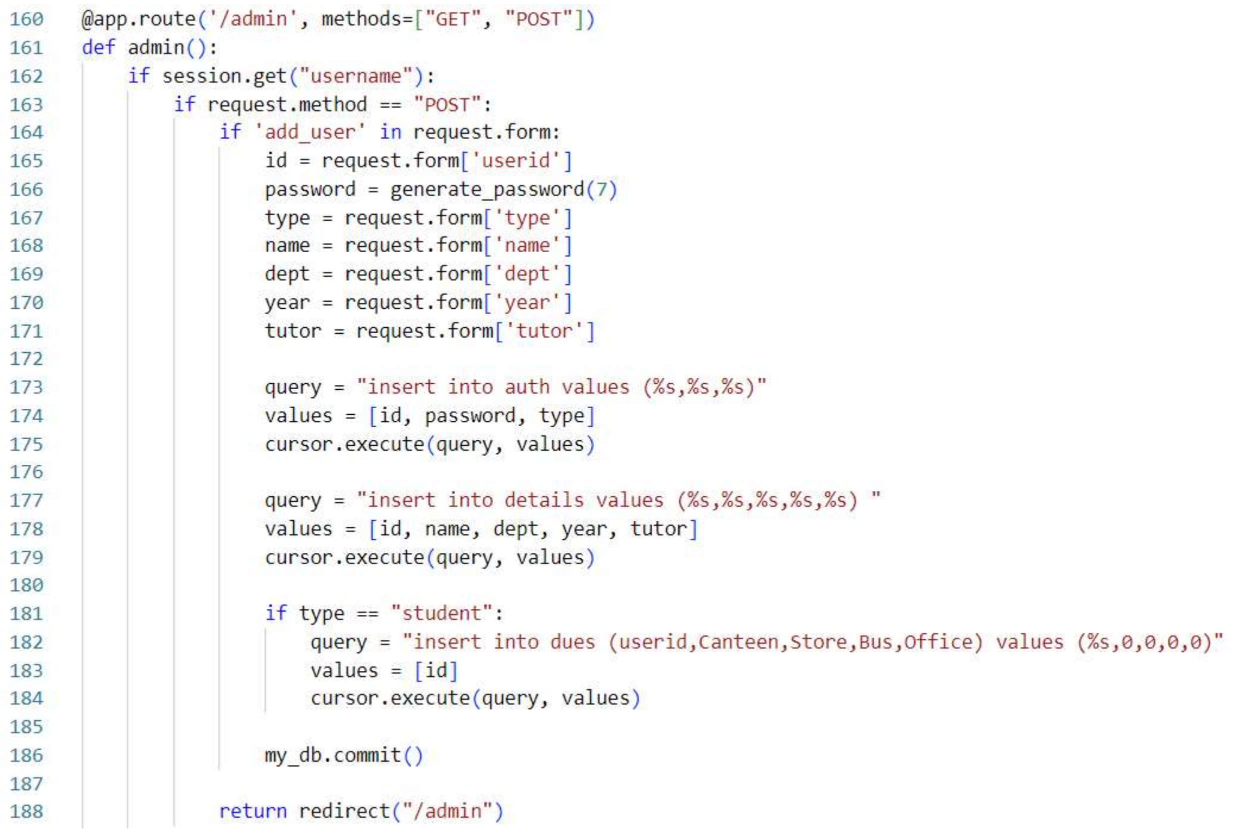
```python
from flask import Flask, render_template, request, redirect, session
import mysql.connector
import random
import string

app = Flask(__name__)
app.secret_key = "pass"

my_db = mysql.connector.connect(
    host="localhost",
    user="root",
    passwd="123456",
    database="cdms"
)
cursor = my_db.cursor()

global allowed_routes
allowed_routes = ["login", "static","logout"]
```

Implements the login and logout routes which deal with user authentication and access management.



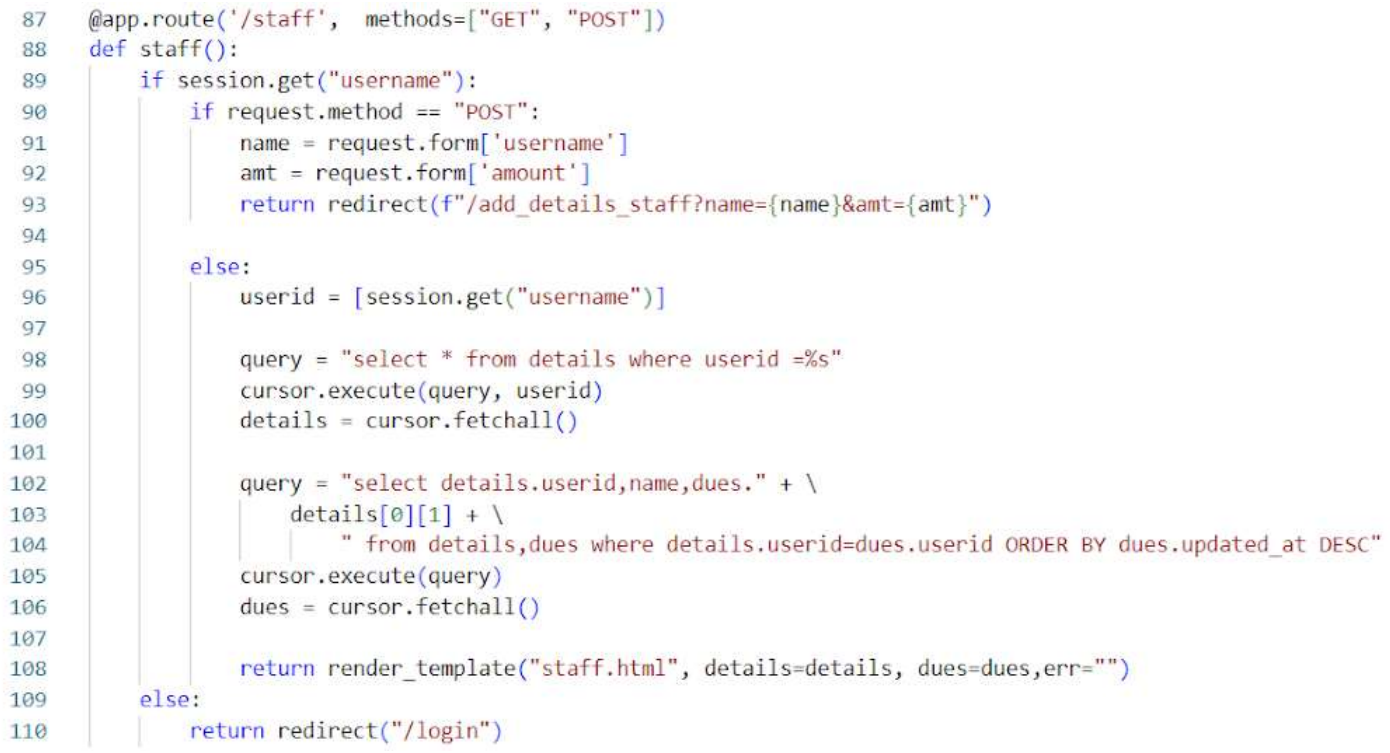
```python
@app.route('/login',  methods=["GET", "POST"])
def login():
    global allowed_routes
    allowed_routes = ["login", "static"]
    if request.method == "POST":
        username = request.form.get("username")
        password = request.form.get("password")

        query = "select type from auth where binary userid = %s and binary password =%s"
        value = [username, password]
        cursor.execute(query, value)
        auth = cursor.fetchall()

        if auth:
            session["authenticated"] = True
            session["username"] = username
            type = auth[0][0]
            print(username,type, " authentication successful")

            if type == "student":
                allowed_routes = ["login","static","logout", "student"]
                return redirect('/student')

            elif type == "staff":
                allowed_routes = ["login", "static","logout",
                                  "staff", "add_details_staff","error"]
                return redirect('/staff')

            elif type == "tutor":
                allowed_routes = ["login", "static","logout", "tutor"]
                return redirect('/tutor')

            elif type == "admin":
                allowed_routes=["login","static","logout","admin","admin_table","edit"]
```

Implements the routes for the admin page and its corresponding functionalities.



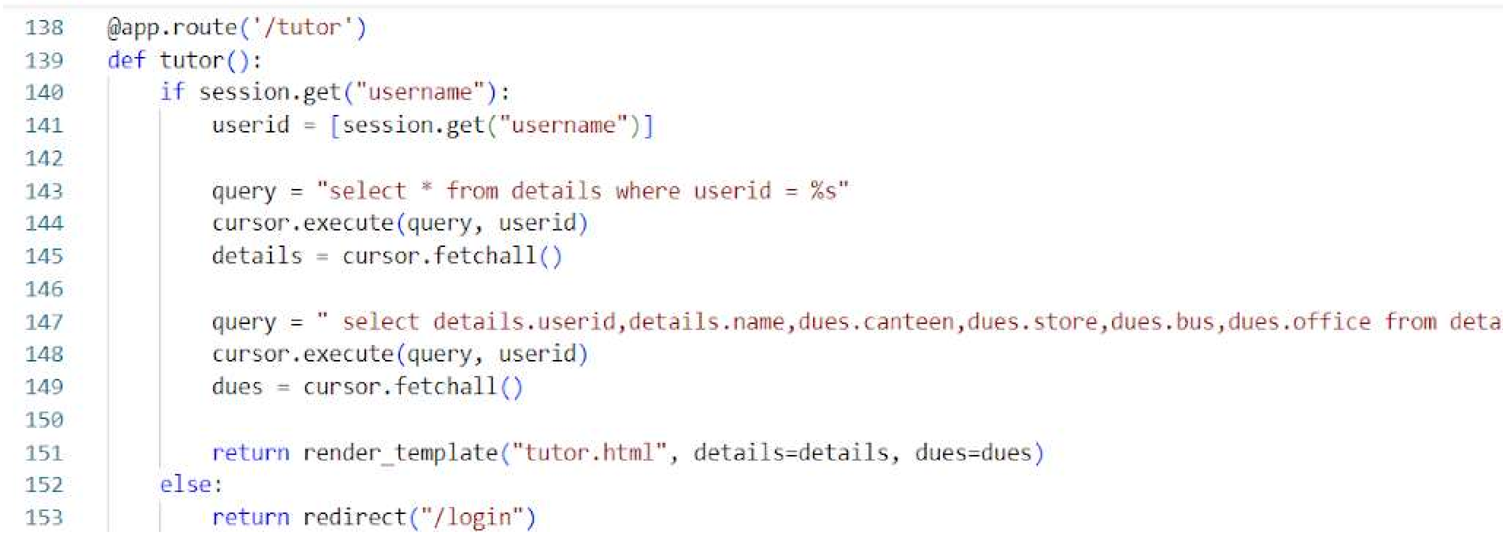
```python
@app.route('/admin', methods=["GET", "POST"])
def admin():
    if session.get("username"):
        if request.method == "POST":
            if 'add_user' in request.form:
                id = request.form['userid']
                password = generate_password(7)
                type = request.form['type']
                name = request.form['name']
                dept = request.form['dept']
                year = request.form['year']
                tutor = request.form['tutor']

                query = "insert into auth values (%s,%s,%s)"
                values = [id, password, type]
                cursor.execute(query, values)

                query = "insert into details values (%s,%s,%s,%s,%s) "
                values = [id, name, dept, year, tutor]
                cursor.execute(query, values)

                if type == "student":
                    query = "insert into dues (userid,Canteen,Store,Bus,Office) values (%s,0,0,0,0)"
                    values = [id]
                    cursor.execute(query, values)

                my_db.commit()

            return redirect("/admin")
```

Implements the routes for the Staff page and its corresponding functionalities.



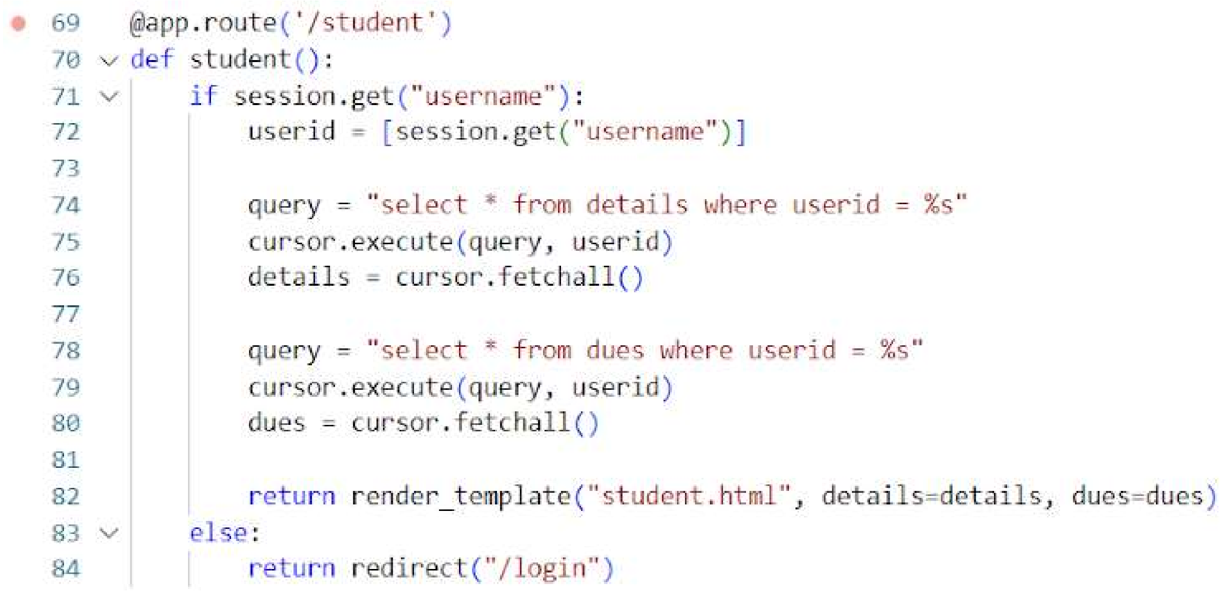
```python
@app.route('/staff',  methods=["GET", "POST"])
def staff():
    if session.get("username"):
        if request.method == "POST":
            name = request.form['username']
            amt = request.form['amount']
            return redirect(f"/add_details_staff?name={name}&amt={amt}")

        else:
            userid = [session.get("username")]

            query = "select * from details where userid =%s"
            cursor.execute(query, userid)
            details = cursor.fetchall()

            query = "select details.userid,name,dues." + \
                details[0][1] + \
                    " from details,dues where details.userid=dues.userid ORDER BY dues.updated_at DESC"
            cursor.execute(query)
            dues = cursor.fetchall()

            return render_template("staff.html", details=details, dues=dues,err="")
    else:
        return redirect("/login")
```

Implements the routes for the Tutor page.



```python
@app.route('/tutor')
def tutor():
    if session.get("username"):
        userid = [session.get("username")]

        query = "select * from details where userid = %s"
        cursor.execute(query, userid)
        details = cursor.fetchall()

        query = " select details.userid,details.name,dues.canteen,dues.store,dues.bus,dues.office from deta
        cursor.execute(query, userid)
        dues = cursor.fetchall()

        return render_template("tutor.html", details=details, dues=dues)
    else:
        return redirect("/login")
```

Implements the routes for the Student page.



```python
@app.route('/student')
def student():
    if session.get("username"):
        userid = [session.get("username")]

        query = "select * from details where userid = %s"
        cursor.execute(query, userid)
        details = cursor.fetchall()

        query = "select * from dues where userid = %s"
        cursor.execute(query, userid)
        dues = cursor.fetchall()

        return render_template("student.html", details=details, dues=dues)
    else:
        return redirect("/login")
```

CHAPTER 5: RESULT AND DISCUSSION

5.1 Result

The College Due Management System was successfully designed, developed, and implemented to address the challenges associated with managing student dues efficiently. The system's functionalities were thoroughly tested, and it demonstrated remarkable performance and usability during the testing phase. The key outcomes and findings from the implementation are presented below:

5.1.1 Centralized Database and Real-Time Updates:

The system's centralized database proved to be highly effective in storing and retrieving student due details from various sectors within the college. Real-time updates allowed students and faculty members to access the most current information regarding outstanding dues promptly.

5.1.2 User-Friendly Interface:

The user-friendly interface of the College Due Management System received positive feedback from users. Both students and faculty members found it intuitive to navigate the system, and little to no training was required for them to utilize its features effectively.

5.1.3 Secure Authentication:

The implementation of robust security measures like secure authentication protocols ensured the protection of sensitive student data from unauthorized access and potential breaches.

5.2 Discussion

5.2.1 Improved Administrative Efficiency:

The College Due Management System has significantly improved administrative efficiency within the college. By automating the process of due management, administrative staff can now focus on more strategic tasks, leading to reduced paperwork and enhanced productivity.

5.2.2 Enhanced Student Experience:

The system's real-time updates have empowered students to take ownership of their dues. This has resulted in fewer instances of overdue dues and smoother semester registration processes. As a result, students now experience a more streamlined and stress-free academic journey.

5.2.3 Transparency and Accountability:

With access to detailed due information, students and faculty members have gained greater transparency into their due status. The system has fostered a sense of accountability, encouraging timely payment of dues and proactive management of financial responsibilities.

5.2.4 Scalability and Future Expansion:

The College Due Management System has been designed with scalability in mind. Its architecture and database are capable of accommodating a growing number of students and courses, ensuring the system's relevance and effectiveness in the future.

5.2.5 User Feedback and Recommendations:

Throughout the implementation phase, user feedback was continuously gathered and analyzed. Based on this feedback, minor adjustments and improvements were made to the system to enhance its usability and functionality further.

5.3 Output

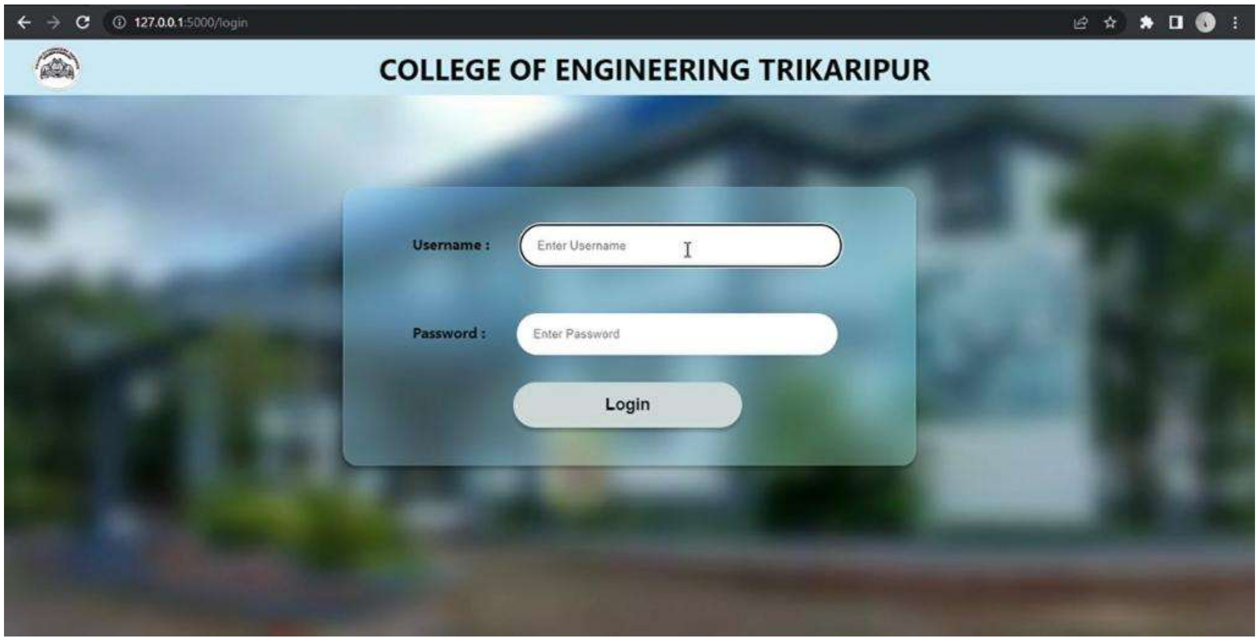
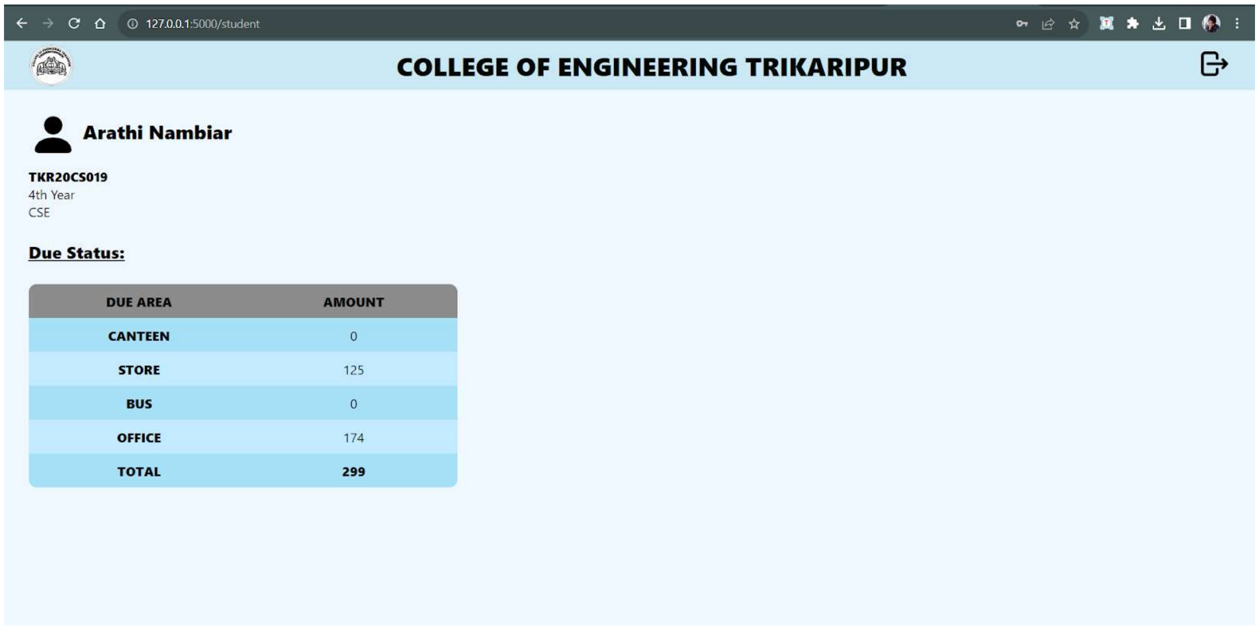


Fig 5.1 Login Page



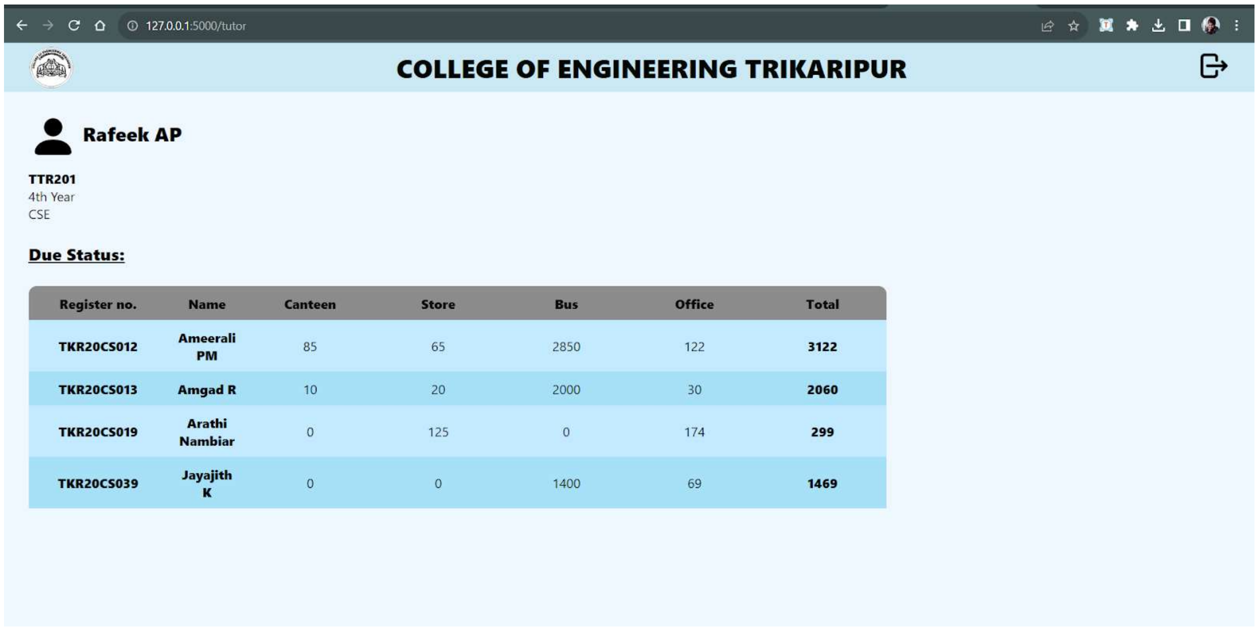
Arathi Nambiar

TKR20CS019
4th Year
CSE

Due Status:

DUE AREA	AMOUNT
CANTEEN	0
STORE	125
BUS	0
OFFICE	174
TOTAL	**299**

Fig 5.2 Student Page



127.0.0.1:5000/tutor

COLLEGE OF ENGINEERING TRIKARIPUR

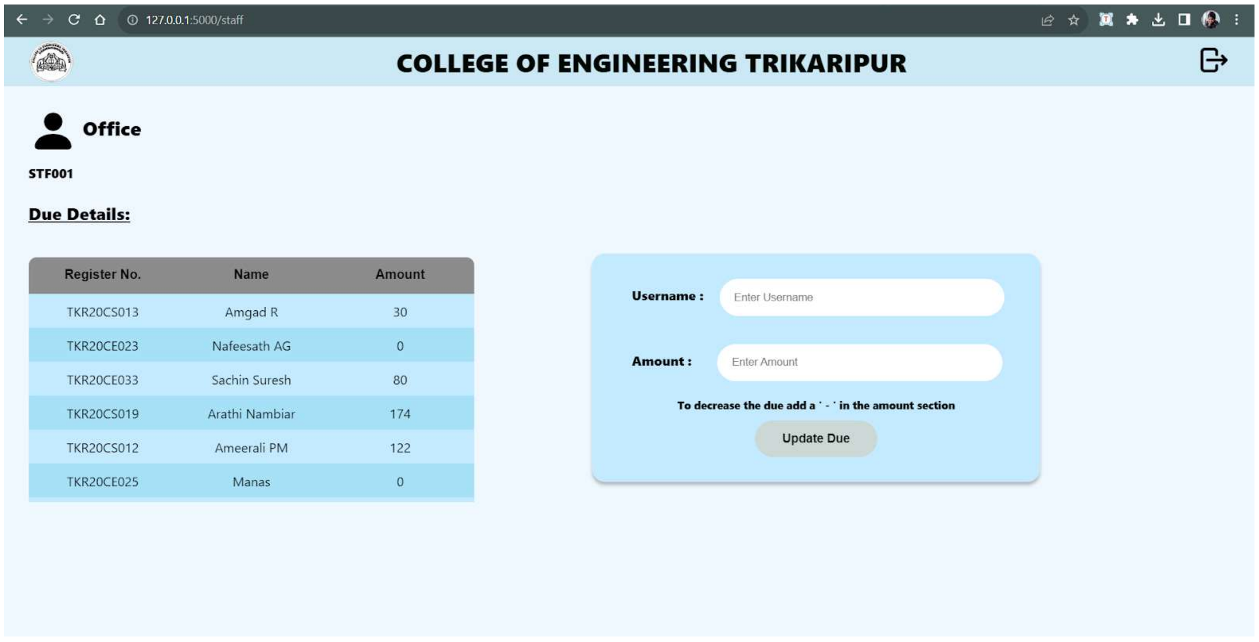
Rafeek AP

TTR201
4th Year
CSE

Due Status:

Register no.	Name	Canteen	Store	Bus	Office	Total
TKR20CS012	Ameerali PM	85	65	2850	122	3122
TKR20CS013	Amgad R	10	20	2000	30	2060
TKR20CS019	Arathi Nambiar	0	125	0	174	299
TKR20CS039	Jayajith K	0	0	1400	69	1469

Fig 5.3 Tutor Page



127.0.0.1:5000/staff

COLLEGE OF ENGINEERING TRIKARIPUR

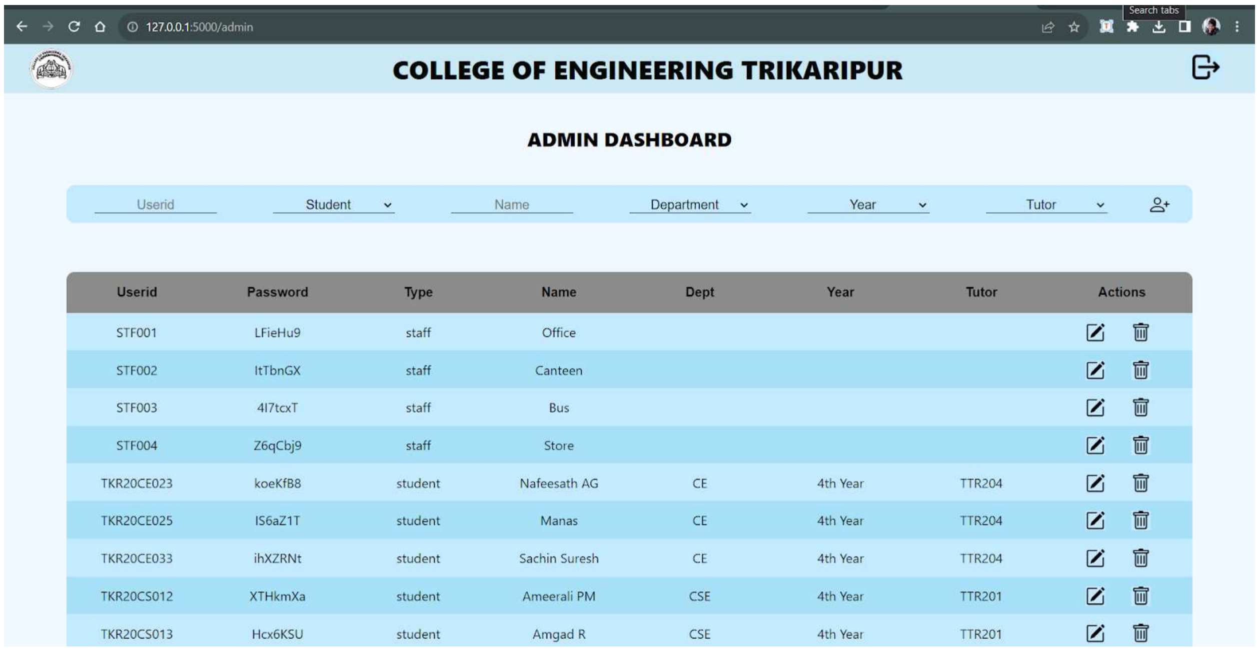
Office

STF001

Due Details:

Register No.	Name	Amount
TKR20CS013	Amgad R	30
TKR20CE023	Nafeesath AG	0
TKR20CE033	Sachin Suresh	80
TKR20CS019	Arathi Nambiar	174
TKR20CS012	Ameerali PM	122
TKR20CE025	Manas	0

Fig 5.4 Staff Page


127.0.0.1:5000/admin

COLLEGE OF ENGINEERING TRIKARIPUR

ADMIN DASHBOARD

Userid	Password	Type	Name	Dept	Year	Tutor	Actions
STF001	LFieHu9	staff	Office				
STF002	ltTbnGX	staff	Canteen				
STF003	4I7tcxT	staff	Bus				
STF004	Z6qCbj9	staff	Store				
TKR20CE023	koeKfB8	student	Nafeesath AG	CE	4th Year	TTR204	
TKR20CE025	IS6aZ1T	student	Manas	CE	4th Year	TTR204	
TKR20CE033	ihXZRNt	student	Sachin Suresh	CE	4th Year	TTR204	
TKR20CS012	XTHkmXa	student	Ameerali PM	CSE	4th Year	TTR201	
TKR20CS013	Hcx6KSU	student	Amgad R	CSE	4th Year	TTR201	

Fig 5.5 Admin Page

CHAPTER 6: CONCLUSION

In our project, we decided to make a simple website which grants a much more promising way to track and keep up with the dues of different students within the college, which the authorities can easily access to view or edit the data.

Our project reduces the process of clearing dues for the authority and introduces a much better way at managing dues of students.

This in turn reduces a lot of paperwork as well as an entire day of approaching different sectors in order to clear dues for each and every student.

REFERENCES

- Building Python web services, Jack Stouffer, Shalabh Aggarwal, Gareth Dwyer, PACKT Publishing Limited, 2018

- Kenneth A Lambert., Fundamentals of Python : First Programs, 2/e, Cengage Publishing, 2016

- Python : https://www.python.org/

- Flask : https://flask.palletsprojects.com/en/2.3.x/

- Jinja : https://jinja.palletsprojects.com/en/3.1.x/