SCHOOL MANAGEMENT SYSTEM

A MINI PROJECT REPORT

Submitted by

RA2211026010016 – Karan Sood

RA2211026010027 – Aditya Nair

Under the guidance of

Dr. T.S. SHINY ANGEL

Associate Professor, Department of Computational Intelligence

In partial fulfilment for the Course

of

21CSC205P – DATABASE MANAGEMENT SYSTEM

in

in the Department of Computational Intelligence

FACULTY OF ENGINEERING AND TECHNOLOGY

SCHOOL OF COMPUTING

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR

MAY 2024

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that this minor project report for the course **21CSC205P – DATABASE MANAGEMENT SYSTEM** entitled in " **SCHOOL MANAGEMENT SYSTEM**" is the bonafide work of **KARAN SOOD (RA2211026010016) , ADITYA NAIR (RA2211026010027) and** who carried out the work under my supervision.

SIGNATURE	**SIGNATURE**
Dr. T.S. SHINY ANGEL	Dr.R.Annie Uthra
Associate Professor,	Professor & Head,
Department of Computational Intelligence	Department of Computational Intelligence
SRM Institute of Science and Technology	SRM Institute of Science and Technology
Kattankulathur	Kattankulathur

ACKNOWLEDGEMENT

We express our heartfelt thanks to our honorable **Vice Chancellor Dr. C. MUTHAMIZHCHELVAN**, for being the beacon in all our endeavors.

We would like to express my warmth of gratitude to our **Registrar Dr. S. Ponnusamy,** for his encouragement.

We express our profound gratitude to our **Dean (College of Engineering and Technology) Dr. T. V.Gopal,** for bringing out novelty in all executions.

We would like to express my heartfelt thanks to Chairperson, School of Computing **Dr. Revathi Venkataraman,** for imparting confidence to complete my course project.

We extend my gratitude to our **Head of the Department, Dr.R.Annie Uthra, Professor & Head, Department of Computational Intelligence** and my Departmental colleagues for their Support.

We are highly thankful to our Course project Faculty **Dr. T.S. SHINY ANGEL , Associate Professor, Department of Computational Intelligence** for her assistance, timely suggestion and guidance throughout the duration of this course project.

Finally, we thank our parents and friends near and dear ones who directly and indirectly contributed to the successful completion of our project. Above all, I thank the almighty for showering his blessings on me to complete my Course project.

ABSTRACT

The School Management System (SMS) is a comprehensive software solution designed to streamline and enhance various administrative and academic processes within educational institutions. This system integrates multiple modules to facilitate efficient management of student records, staff information, academic schedules, attendance tracking, and resource allocation. Through the SMS, administrators can easily register new students, maintain detailed student profiles, and manage enrollment and admissions processes. Additionally, the system enables automated tracking of attendance, grading, and academic performance, providing educators with valuable insights to support student success. SMS also facilitates communication between stakeholders, allowing for seamless interaction between administrators, teachers, students, and parents. With its user-friendly interface and robust functionality, the School Management System serves as a central hub for organizing, analyzing, and optimizing the myriad operations within educational institutions, ultimately fostering a conducive environment for learning and growth.

TABLE OF CONTENTS

CHAPTER NO	CONTENTS	PAGE NO
1	**INTRODUCTION**	
	1.1 Motivation	**03**
	1.2 Objective	**03-04**
	1.3 Problem Statements	**04**
	1.4 Challenges	**04-05**
2	**REQUIREMENT ANALYSIS**	**06**
3	**ARCHITECTURE & DESIGN**	**07**
4	**IMPLEMENTATION**	**08-20**
5	**EXPERIMENT RESULTS & ANALYSIS**	**21-25**
6	**CONCLUSION**	**26**
7	**REFERENCES**	**27**

1. INTRODUCTION

1.1 Motivation

Implementing a School Management System is a transformative endeavor that aims to enhance the efficiency, organization, and overall effectiveness of educational institutions. The motivation behind developing such a system lies in addressing various challenges faced by schools and administrators, while simultaneously leveraging technological advancements to streamline processes and improve outcomes.

Firstly, a School Management System serves to centralize and automate administrative tasks, reducing the burden on school staff and administrators. By digitizing processes such as attendance tracking, student enrollment, fee management, and academic record-keeping, the system eliminates manual paperwork and minimizes errors, leading to significant time and resource savings.

Moreover, the system facilitates better communication and collaboration among stakeholders, including teachers, students, parents, and school management. Through features such as online portals, messaging systems, and real-time updates on academic progress and events, it fosters transparency, engagement, and active involvement in the educational journey.

1.2 OBJECTIVE

The objective of a school management system is to streamline and automate various administrative and academic tasks within an educational institution. This system serves as a centralized platform for managing student information, staff records, academic activities, and other essential functions.

1. **Student Information Management**: The system facilitates the efficient management of student data, including enrollment details, academic records, attendance, and personal information.

2. **Staff Administration**: It provides tools for managing staff information such as employment records, qualifications, schedules, and payroll processing.

3. **Academic Management:** The system supports academic activities such as course scheduling, grading, exam management, and curriculum planning.

4. **Attendance Monitoring**: It enables real-time tracking of student attendance, helping educators and administrators identify patterns and take timely interventions as needed.

5. **Financial Management**: The system helps in managing finances by handling fee payments, generating invoices, tracking expenses, and budgeting.

1.3 PROBLEM STATEMENTS

The school management system aims to streamline various administrative tasks within an educational institution, enhancing efficiency and organization. This system typically encompasses functionalities such as student enrollment, attendance tracking, grading, scheduling, and communication between stakeholders.

In this system, administrators can manage student information, including personal details, academic records, and contact information. They can also register new students, update existing records, and handle student transfers or withdrawals.

Attendance tracking is a crucial feature of the system, allowing teachers to record students' presence or absence during classes. This data is then used for generating attendance reports, identifying trends, and addressing potential issues related to student attendance.

Grading functionality enables educators to input and calculate student grades for various assignments, tests, and exams. The system may also support the creation of grade books, report cards, and transcripts, providing a comprehensive overview of student performance.

1.4 CHALLENGES

Implementing a school management system presents several challenges, ranging from technical complexities to ensuring smooth administrative operations and user satisfaction:

1**. Database Management**: Designing an efficient database schema to store diverse information

such as student details, attendance records, grades, staff information, etc., while ensuring data integrity, normalization, and scalability can be challenging.

2. **User Interface Design**: Creating an intuitive and user-friendly interface for administrators, teachers, students, and parents requires thorough understanding of user requirements, accessibility considerations, and design principles to ensure smooth navigation and usability.

3. **Security:** Safeguarding sensitive data such as student records, financial information, and personal details against unauthorized access, data breaches, and cyber threats is crucial. Implementing robust authentication, authorization, and encryption mechanisms is essential to maintain data security.

4. **Attendance Tracking**: Developing a reliable system to accurately track student attendance, manage absence records, and generate attendance reports in real-time can be challenging, especially in large educational institutions with multiple classes and varying attendance policies.

5. **Integration with External Systems:** Integrating the school management system with external systems such as student information systems (SIS), learning management systems (LMS), financial management software, and communication platforms requires seamless data exchange, API integration, and interoperability considerations.

REQUIREMENT ANALYSIS

A school management system aims to streamline and optimize various administrative and academic processes within an educational institution. It typically encompasses a wide range of functionalities to facilitate efficient management of student data, academic resources, staff management, and communication among stakeholders. Below is a requirement analysis for a school management system:

1. **Student Information Management**: The system should allow administrators to store and manage comprehensive student information, including personal details, academic records, attendance, disciplinary records, and contact information.

2. **Staff Management**: It should enable the management of staff data, including teachers, administrative staff, and other personnel. This includes storing information such as contact details, employment history, qualifications, and roles within the institution.

3. **Course and Curriculum Management**: The system should support the management of courses, subjects, and curriculum details. This includes defining course structures, assigning teachers to courses, and managing curriculum updates.

4. **Attendance Management**: There should be functionality to record and track student attendance for each class or session. The system should provide the ability to generate attendance reports and notify stakeholders of any irregularities.

5. **Grading and Academic Performance Tracking**: The system should allow teachers to input grades and track student academic performance over time. It should support the calculation of GPA, class ranks, and generation of academic transcripts.

6. **Communication and Collaboration**: The system should facilitate communication between various stakeholders, including students, parents, teachers, and administrators. This could include features such as messaging, announcements, and forums.

7. **Admissions and Enrollment**: The system should support the admissions process, including online application submissions, document verification, and enrollment management. It should also manage student registration and class assignments.

By addressing these requirements, a school management system can effectively streamline operations, improve communication, and enhance overall efficiency within the educational institution.

ARCHITECTURE AND DESIGN

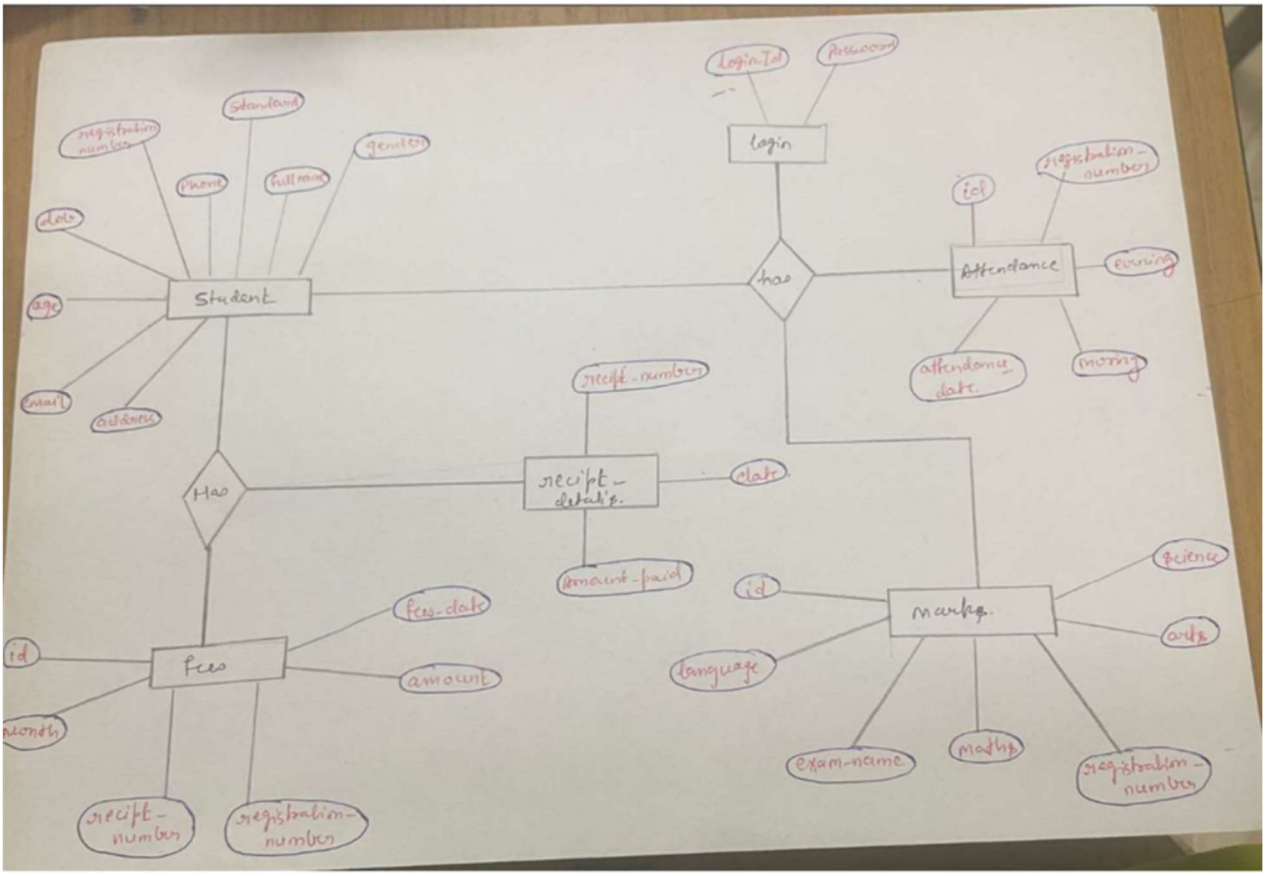


Fig 1

IMPLEMENTATION

SQL TABLES

show tables;

```
+---------------------+
| Tables_in_school_db |
+---------------------+
| attendance          |
| fees                |
| login               |
| marks               |
| receipt_details     |
| student             |
+---------------------+
```

ATTENDANCE TABLE STRUCTURE

```
desc attendance;
+---------------------+--------------+------+-----+---------+----------------+
| Field               | Type         | Null | Key | Default | Extra          |
+---------------------+--------------+------+-----+---------+----------------+
| id                  | int          | NO   | PRI | NULL    | auto_increment |
| registration_number | varchar(255) | YES  |     | NULL    |                |
| attendance_date     | date         | YES  |     | NULL    |                |
| morning             | varchar(10)  | YES  |     | NULL    |                |
| evening             | varchar(10)  | YES  |     | NULL    |                |
+---------------------+--------------+------+-----+---------+----------------+
```

DATA IN ATTENDANCE TABLE

```
select * from attendance;
+----+---------------------+-----------------+---------+---------+
| id | registration_number | attendance_date | morning | evening |
+----+---------------------+-----------------+---------+---------+
|  1 | 1                   | 2024-02-08      | ABSENT  | ABSENT  |
|  2 | 1                   | 2024-03-28      | PRESENT | ABSENT  |
+----+---------------------+-----------------+---------+---------+
```

FEES TABLE STRUCTURE

Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	auto_increment
receipt_number	int	YES		NULL	
registration_number	varchar(255)	YES		NULL	
month	varchar(255)	YES		NULL	
amount	decimal(10,2)	YES		NULL	
fees_date	date	YES		NULL	

DATA IN FEES TABLE

id	receipt_number	registration_number	month	amount	fees_date
1	1	1	February	4500.00	2024-02-08
2	2	1	March	4600.00	2024-03-27

LOGIN TABLE STRUCTURE

Field	Type	Null	Key	Default	Extra
user_id	int	NO	PRI	NULL	
password	varchar(255)	NO		NULL	

MARKS TABLE STRUCTURE

Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	auto_increment
registration_number	varchar(255)	YES		NULL	
exam_name	varchar(255)	YES		NULL	
language	int	YES		NULL	
maths	int	YES		NULL	

```
| science            | int         | YES |     | NULL    |              |
| arts               | int         | YES |     | NULL    |              |
+--------------------+-------------+-----+-----+---------+--------------+
```

DATA IN MARKS TABLE

```
+----+--------------------+-----------+----------+-------+---------+------+
| id | registration_number | exam_name | language | maths | science | arts |
+----+--------------------+-----------+----------+-------+---------+------+
|  3 | 2                  |           |       60 |    70 |      20 |   50 |
|  4 | 3                  |           |       60 |    90 |      40 |   60 |
|  5 | 5                  |           |       80 |    80 |      95 |   60 |
+----+--------------------+-----------+----------+-------+---------+------+
```

STUDENT TABLE STRUCTURE

```
+--------------------+-------------+-----+-----+---------+---------------+
| Field              | Type        | Null | Key | Default | Extra         |
+--------------------+-------------+-----+-----+---------+---------------+
| registration_number | int         | NO  | PRI | NULL    | auto_increment |
| full_name          | varchar(255) | NO  |     | NULL    |               |
| gender             | varchar(10)  | NO  |     | NULL    |               |
| dob                | date        | NO  |     | NULL    |               |
| age                | int         | NO  |     | NULL    |               |
| address            | text        | YES |     | NULL    |               |
| phone              | varchar(15)  | YES |     | NULL    |               |
| email              | varchar(255) | YES |     | NULL    |               |
| standard           | varchar(10)  | YES |     | NULL    |               |
+--------------------+-------------+-----+-----+---------+---------------+
```

DATA IN STUDENT TABLE

```
+--------------------+------------+--------+-----------+-----+-------------------+-----------+-----------
-----------+----------+
| registration_number | full_name  | gender | dob       | age | address           | phone     | email
 | standard |
+--------------------+------------+--------+-----------+-----+-------------------+-----------+-----------
-----------+----------+
|                  1 | KARAN SOOD | MALE   | 2004-04-13 |  19 | SANASSI 402       | 7011795472
| KS6102@SRMIST.EDU.IN | 01       |
|                  2 | Aditya Nair | MALE   | 2004-09-11 |  19 | udaipur,rajasthan | 7011795472 |
an0995@srmist.edu.in | 10       |
+--------------------+------------+--------+-----------+-----+-------------------+-----------+-----------
----------
```

IMPLEMENTATION

Source Code

```
from datetime import datetime
from PyQt6.QtCore import *
from PyQt6.QtGui import *
from PyQt6.QtWidgets import *
import sys
from PyQt6.uic import loadUiType
import mysql.connector as con
from PyQt6.QtGui import QPainter, QPixmap
from PyQt6.QtPrintSupport import QPrinter, QPrintDialog, QPrintPreviewDialog
ui, _ = loadUiType('schl_mgmt_app/qt_ui/school.ui')
class AspectRatioSizeGrip(QSizeGrip):
    def mouseMoveEvent(self, event):
        window = self.window()
        diff = event.globalPosition().toPoint() - window.geometry().topLeft()
        newSize = diff.boundedTo(
            window.maximumSize()).expandedTo(window.minimumSize())
        ratio = window.width() / window.height()
        newSize.setHeight(newSize.width() / ratio)
        window.resize(newSize)
class MainApp(QMainWindow, ui):
    def __init__(self):
        QMainWindow.__init__(self)
        self.setupUi(self)
        self.setFixedSize(self.size())
        self.tabWidget.setCurrentIndex(0)
        self.tabWidget.tabBar().setVisible(False)
        self.menubar.setVisible(False)
        self.b01.clicked.connect(self.login)
        self.menu_01_01.triggered.connect(self.show_add_new_student)
        self.b12.clicked.connect(self.save_student_details)
        self.b11.clicked.connect(
            lambda: self.calculateAge(self.tb13, self.tb14))

        self.menu_01_02.triggered.connect(self.show_edit_or_delete_student)
        self.cb21.currentIndexChanged.connect(self.fetch_student_details)
        self.b21.clicked.connect(self.update_student_details)
        self.b22.clicked.connect(self.delete_student_details)
        self.b23.clicked.connect(
            lambda: self.calculateAge(self.tb22, self.tb23))

        self.menu_02_01.triggered.connect(self.show_add_or_edit_marks)
        self.b31.clicked.connect(self.save_marks_details)
        self.cb33.currentIndexChanged.connect(self.marks_fetch_exams)
        self.b32.clicked.connect(self.fetch_exam_marks)
        self.b33.clicked.connect(self.update_exam_marks)
        self.b34.clicked.connect(self.delete_exam_marks)
        self.menu_03_01.triggered.connect(self.show_attendance)
        self.b41.clicked.connect(self.save_attendance_details)
        self.cb42.currentIndexChanged.connect(self.attendance_fetch_dates)
        self.b44.clicked.connect(self.fetch_attendance_details)
        self.b42.clicked.connect(self.update_attendance_details)
        self.b43.clicked.connect(self.delete_attendance_details)
        self.menu_04_01.triggered.connect(self.show_fees)
        self.b51.clicked.connect(self.save_fees_details)
        self.b81.clicked.connect(self.print_receipt)
        self.b81.clicked.connect(lambda: self.tabWidget.setCurrentIndex(1))
        self.b82.clicked.connect(lambda: self.tabWidget.setCurrentIndex(1))
        self.cb52.currentIndexChanged.connect(self.fetch_receipt_details)
        self.b52.clicked.connect(self.update_fees_details)
        self.b53.clicked.connect(self.delete_fees_details)
        self.menu_05_01.triggered.connect(self.show_report)
        self.menu_05_02.triggered.connect(self.show_report)
        self.menu_05_03.triggered.connect(self.show_report)
        self.menu_05_04.triggered.connect(self.show_report)
        self.menu_06_01.triggered.connect(self.logout)
    def login(self):
        un = self.tb01.text()
        pw = self.tb02.text()
        if (un == "admin" and pw == "admin"):
            self.menubar.setVisible(True)
            self.tabWidget.setCurrentIndex(1)
        else:
            QMessageBox.information(
                self, "School Management System", "Invalid Credentials! Try Again !")
```

```
            self.l01.setText("Invalid Credentials !!")
    def show_add_new_student(self):
        self.tabWidget.setCurrentIndex(2)
        self.fill_registration_number()
    def fill_registration_number(self):
        try:
            rn = 0
            mydb = con.connect(host="localhost", user="root",
                               password="user", db="school_db")
            cursor = mydb.cursor(buffered=True, dictionary=True)
            cursor.execute("select * from student")
            result = cursor.fetchall()
            if result:
                for stud in result:
                    rn += 1
            self.tb11.setText(str(rn+1))
        except con.Error as e:
            print("Error Occurred in Connecting to school_db" + str(e))
    def save_student_details(self):
        try:
            mydb = con.connect(host="localhost", user="root",
                               password="user", db="school_db")
            cursor = mydb.cursor(buffered=True, dictionary=True)
            cursor.execute("SELECT * FROM student")
            registration_number = self.tb11.text()
            full_name = self.tb12.text()
            dob = datetime.strptime(self.tb13.text(), "%d-%m-%Y").strftime("%Y-%m-%d")
            age = self.tb14.text()
            phone = self.tb15.text()
            email = self.tb16.text()
            address = self.mtb11.toPlainText()
            gender = self.cb11.currentText()
            standard = self.cb12.currentText()
            qry = "INSERT INTO student (registration_number, full_name, gender, dob, age, address, phone, email, standard) VALUES
(%s, %s, %s, %s, %s, %s, %s, %s, %s)"
            value = (registration_number, full_name, gender,
                     dob, age, address, phone, email, standard)
            cursor.execute(qry, value)
            mydb.commit()
            self.l11.setText("Student Details Saved Successfully !!")
            QMessageBox.information(
                self, "School Management System", "Student Details Saved Successfully !!")
            self.tb11.setText("")
            self.tb12.setText("")
            self.tb13.setDate(QDate())
            self.tb14.setText("")
            self.tb15.setText("")
            self.tb16.setText("")
            self.mtb11.setText("")
            self.l11.setText("")
            self.tabWidget.setCurrentIndex(1)
        except con.Error as e:
            self.l11.setText("Error! Could not Save Student Details ! ")
            print("Error Occurred in Connecting to school_db " + str(e))
    def calculateAge(self, dob, set_age):
        dob_text = dob.text()
        try:
            dob_date = datetime.strptime(dob_text, "%d-%m-%Y").date()
            today = QDate.currentDate().toPyDate()
            age = today.year - dob_date.year - \
                ((today.month, today.day) < (dob_date.month, dob_date.day))
            set_age.setText(str(age))  # set age
        except ValueError:
            print("Invalid DOB format")
    def show_edit_or_delete_student(self):
        self.tabWidget.setCurrentIndex(3)
        self.fetch_registration_number()
    def fetch_registration_number(self):
        try:
            self.cb21.clear()
            mydb = con.connect(host="localhost", user="root",
                               password="user", db="school_db")
            cursor = mydb.cursor(buffered=True, dictionary=True)
            cursor.execute("select * from student")
            result = cursor.fetchall()
            if result:
                for stud in result:
                    self.cb21.addItem(str(stud['registration_number']))
        except con.Error as e:
            print("Error Occurred in Connecting to school_db " + str(e))
    def fetch_student_details(self):
        try:
            mydb = con.connect(host="localhost", user="root",
                               password="user", db="school_db")
```

```
            cursor = mydb.cursor(buffered=True, dictionary=True)
            cursor.execute(
                "select * from student where registration_number = '"+self.cb21.currentText()+"'")
            result = cursor.fetchall()
            if result:
                for stud in result:
                    self.tb21.setText(str(stud['full_name']))
                    self.tb22.setDate(QDate.fromString(
                        str(stud['dob']), "yyyy-mm-dd"))
                    self.tb23.setText(str(stud['age']))
                    self.tb24.setText(str(stud['phone']))
                    self.tb25.setText(str(stud['email']))
                    self.mtb21.setText(str(stud['address']))
                    self.cb22.setCurrentText(str(stud['gender']))
                    self.cb23.setCurrentText(str(stud['standard']))
        except con.Error as e:
            print("Error Occurred in Connecting to school_db " + str(e))
    def update_student_details(self):
        try:
            mydb = con.connect(host="localhost", user="root",
                               password="user", db="school_db")
            cursor = mydb.cursor(buffered=True, dictionary=True)
            registration_number = self.cb21.currentText()
            full_name = self.tb21.text()
            dob = self.tb22.text()
            age = self.tb23.text()
            phone = self.tb24.text()
            email = self.tb25.text()
            address = self.mtb21.toPlainText()
            gender = self.cb22.currentText()
            standard = self.cb23.currentText()
            qry = "UPDATE student set full_name = '" + full_name + "', gender = '" + gender + "', dob = '" + dob + "', age = '" +
age + "', address = '" + \
                address + "', phone = '" + phone + "', email = '" + email + "', standard = '" + \
                standard + "' where registration_number = '" + registration_number + "'"
            cursor.execute(qry)
            mydb.commit()
            self.l21.setText("Student Details Modified Successfully !!")
            QMessageBox.information(
                self, "School Management System", "Student Details Modified Successfully !!")
            self.cb21.clear()
            self.tb11.setText("")
            self.tb12.setText("")
            self.tb13.setDate(QDate())
            self.tb14.setText("")
            self.tb15.setText("")
            self.tb16.setText("")
            self.mtb11.setText("")
            self.l21.setText("")
            self.tabWidget.setCurrentIndex(1)
        except con.Error as e:
            self.l21.setText("Error! Could not Modify Student Details ! ")
            print("Error Occurred in Connecting to school_db " + str(e))
    def delete_student_details(self):
        query = QMessageBox.question(
            self, "Delete", "Are you sure you\nWant to delete This Student details?", QMessageBox.StandardButton.Yes |
QMessageBox.StandardButton.No)
        if query == QMessageBox.StandardButton.Yes:
            try:
                mydb = con.connect(host="localhost", user="root",
                                   password="user", db="school_db")
                cursor = mydb.cursor(buffered=True, dictionary=True)
                registration_number = self.cb21.currentText()
                qry = "delete from student where registration_number = '" + registration_number + "'"
                cursor.execute(qry)
                mydb.commit()
                self.l21.setText("Student Details Deleted Successfully !!")
                QMessageBox.information(
                    self, "School Management System", "Student Details Deleted Successfully !!")
                self.cb21.clear()
                self.l21.setText("")
                self.tabWidget.setCurrentIndex(1)
            except con.Error as e:
                self.l21.setText("Error! Could not Modify Student Details ! ")
                print("Error Occurred in Connecting to school_db " + str(e))
    def show_add_or_edit_marks(self):
        self.tabWidget.setCurrentIndex(4)
        self.marks_fetch_registration_number()
    def marks_fetch_registration_number(self):
        try:
            self.cb31.clear()
            self.cb33.clear()
            mydb = con.connect(host="localhost", user="root",
                               password="user", db="school_db")
```

```
            cursor = mydb.cursor(buffered=True, dictionary=True)
            cursor.execute("select * from student")
            result = cursor.fetchall()
            if result:
                for stud in result:
                    self.cb31.addItem(str(stud['registration_number']))
                    self.cb33.addItem(str(stud['registration_number']))
        except con.Error as e:
            print("Error Occurred in Connecting to school_db " + str(e))
    def save_marks_details(self):
        try:
            mydb = con.connect(host="localhost", user="root",
                               password="user", db="school_db")
            cursor = mydb.cursor(buffered=True, dictionary=True)
            registration_number = self.cb31.currentText()
            exam_name = self.cb32.currentText()
            language = self.tb31.text()
            maths = self.tb32.text()
            science = self.tb33.text()
            arts = self.tb34.text()
            qry = "INSERT INTO marks (registration_number, exam_name, language, maths, science, arts) VALUES (%s, %s, %s, %s, %s,
%s)"
            value = (registration_number, exam_name,
                     language, maths, science, arts)
            cursor.execute(qry, value)
            mydb.commit()
            self.l31.setText("Marks Saved Successfully !!")
            QMessageBox.information(
                self, "School Management System", "Marks Saved Successfully !!")
            self.cb31.currentText()
            self.cb32.setCurrentIndex(-1)
            self.tb31.setText("")
            self.tb32.setText("")
            self.tb33.setText("")
            self.tb34.setText("")
            self.l31.setText("")
            self.tabWidget.setCurrentIndex(1)
        except con.Error as e:
            self.l31.setText("Error! Marks Not Saved ! ")
            print("Error Occurred in Connecting to school_db " + str(e))
    def marks_fetch_exams(self):
        try:
            self.cb34.clear()
            registration_number = self.cb33.currentText()
            mydb = con.connect(host="localhost", user="root",
                               password="user", db="school_db")
            cursor = mydb.cursor(buffered=True, dictionary=True)
            cursor.execute(
                "select * from marks where registration_number = '" + registration_number + "'")
            result = cursor.fetchall()
            if result:
                for stud in result:
                    self.cb34.addItem(stud.get('exam_name', ''))
        except con.Error as e:
            print("Error Occurred in Connecting to school_db " + str(e))
    def fetch_exam_marks(self):
        try:
            registration_number = self.cb33.currentText()
            exam_name = self.cb34.currentText()
            mydb = con.connect(host="localhost", user="root",
                               password="user", db="school_db")
            cursor = mydb.cursor(buffered=True, dictionary=True)
            cursor.execute(
                "select * from marks where registration_number = '" + registration_number + "' and exam_name='" + exam_name + "'")
            result = cursor.fetchall()
            if result:
                for stud in result:
                    self.tb35.setText(str(stud['language']))
                    self.tb36.setText(str(stud['maths']))
                    self.tb37.setText(str(stud['science']))
                    self.tb38.setText(str(stud['arts']))
        except con.Error as e:
            print("Error Occurred in Connecting to school_db " + str(e))
    def update_exam_marks(self):
        try:
            mydb = con.connect(host="localhost", user="root",
                               password="user", db="school_db")
            cursor = mydb.cursor(buffered=True, dictionary=True)
            registration_number = self.cb33.currentText()
            exam_name = self.cb34.currentText()
            language = self.tb35.text()
            maths = self.tb36.text()
            science = self.tb37.text()
            arts = self.tb38.text()
```

```
            qry = "UPDATE marks set language = '" + language + "', maths = '" + maths + "', science = '" + science + \
                "', arts = '" + arts + "' where registration_number = '" + \
                registration_number + "' and exam_name='" + exam_name + "'"
            cursor.execute(qry)
            mydb.commit()
            self.l32.setText("Marks Modified Successfully !!")
            QMessageBox.information(
                self, "School Management System", "Marks Modified Successfully !!")
            self.tb35.setText("")
            self.tb36.setText("")
            self.tb37.setText("")
            self.tb38.setText("")
            self.l32.setText("")
            self.tabWidget.setCurrentIndex(1)
        except con.Error as e:
            self.l21.setText("Error! Modification Unsuccessful ! ")
            print("Error Occurred in Connecting to school_db " + str(e))
    def delete_exam_marks(self):
        query = QMessageBox.question(
            self, "Delete", "Are you sure you\nWant to delete This Exam Marks?", QMessageBox.StandardButton.Yes |
QMessageBox.StandardButton.No)
        if query == QMessageBox.StandardButton.Yes:
            try:
                mydb = con.connect(host="localhost", user="root",
                                   password="user", db="school_db")
                cursor = mydb.cursor(buffered=True, dictionary=True)
                registration_number = self.cb33.currentText()
                exam_name = self.cb34.currentText()
                qry = "delete from marks where registration_number = '" + \
                    registration_number + "' and exam_name='" + exam_name + "'"
                cursor.execute(qry)
                mydb.commit()
                self.l32.setText("Marks Deleted Successfully !!")
                QMessageBox.information(
                    self, "School Management System", "Marks Deleted Successfully !!")
                self.tb35.setText("")
                self.tb36.setText("")
                self.tb37.setText("")
                self.tb38.setText("")
                self.l32.setText("")
                self.tabWidget.setCurrentIndex(1)
            except con.Error as e:
                self.l32.setText("Error! Could not Delete Marks ! ")
                print("Error Occurred in Connecting to school_db " + str(e))
    def show_attendance(self):
        self.tabWidget.setCurrentIndex(5)
        self.attendance_fetch_registration_number()
        self.tb41.setDate(QDate.currentDate())
        self.tb42.setCurrentIndex(-1)
        self.tb43.setCurrentIndex(-1)
        self.tb44.setCurrentIndex(-1)
        self.tb45.setCurrentIndex(-1)
    def attendance_fetch_registration_number(self):
        try:
            self.cb41.clear()
            self.cb42.clear()
            mydb = con.connect(host="localhost", user="root",
                               password="user", db="school_db")
            cursor = mydb.cursor(buffered=True, dictionary=True)
            cursor.execute("select * from student")
            result = cursor.fetchall()
            if result:
                for stud in result:
                    self.cb41.addItem(str(stud['registration_number']))
                    self.cb42.addItem(str(stud['registration_number']))
        except con.Error as e:
            print("Error Occurred in Connecting to school_db " + str(e))
    def save_attendance_details(self):
        try:
            mydb = con.connect(host="localhost", user="root",
                               password="user", db="school_db")
            cursor = mydb.cursor(buffered=True, dictionary=True)
            registration_number = self.cb41.currentText()
            attendance_date = self.tb41.date().toString("yyyy-MM-dd")
            morning = self.tb42.currentText()
            evening = self.tb43.currentText()
            qry = "INSERT INTO attendance (registration_number, attendance_date, morning, evening) VALUES (%s, %s, %s, %s)"
            value = (registration_number, attendance_date, morning, evening)
            cursor.execute(qry, value)
            mydb.commit()
            self.l41.setText("Attendance Saved Successfully !!")
            QMessageBox.information(
                self, "School Management System", "Attendance Saved Successfully !!")
            self.tb42.setCurrentIndex(0)
```

```
            self.tb43.setCurrentIndex(0)
            self.l41.setText("")
            self.tabWidget.setCurrentIndex(1)
        except con.Error as e:
            self.l41.setText("Error! Attendance Not Saved ! ")
            print("Error Occurred in Connecting to school_db " + str(e))
    def attendance_fetch_dates(self):
        try:
            self.cb43.clear()
            registration_number = self.cb42.currentText()

            # Establish database connection
            mydb = con.connect(host="localhost", user="root", password="user", db="school_db")
            cursor = mydb.cursor(buffered=True, dictionary=True)

        # Fetch attendance dates for the given registration number
            query = "SELECT DISTINCT attendance_date FROM attendance WHERE registration_number = %s"
            cursor.execute(query, (registration_number,))
            result = cursor.fetchall()

            # Populate combo box with attendance dates
            if result:
                for row in result:
                    self.cb43.addItem(str(row['attendance_date']))

        except con.Error as e:
            print("Error occurred while fetching attendance dates:", str(e))
        finally:
            # Close database connection
            if mydb:
                mydb.close()
    def fetch_attendance_details(self):
        try:
            registration_number = self.cb42.currentText()
            attendance_date = self.cb43.currentText()
            mydb = con.connect(host="localhost", user="root",
                               password="user", db="school_db")
            cursor = mydb.cursor(buffered=True, dictionary=True)
            cursor.execute(
                "select * from attendance where registration_number = '" + registration_number + "' and attendance_date='" +
attendance_date + "'")
            result = cursor.fetchall()
            if result:
                for stud in result:
                    self.tb44.setCurrentText(str(stud['morning']))
                    self.tb45.setCurrentText(str(stud['evening']))
        except con.Error as e:
            print("Error Occurred in Connecting to school_db " + str(e))
    def update_attendance_details(self):
        try:
            mydb = con.connect(host="localhost", user="root",
                               password="user", db="school_db")
            cursor = mydb.cursor(buffered=True, dictionary=True)
            registration_number = self.cb42.currentText()
            attendance_date = self.cb43.currentText()
            morning = self.tb44.currentText()
            evening = self.tb45.currentText()
            qry = "UPDATE attendance set morning = '" + morning + "', evening = '" + evening + "' where registration_number = '" + \
                registration_number + "' and attendance_date='" + attendance_date + "'"
            cursor.execute(qry)
            mydb.commit()
            self.l42.setText("Attendance Modified Successfully !!")
            QMessageBox.information(
                self, "School Management System", "Attendance Modified Successfully !!")
            self.tb44.setCurrentIndex(-1)
            self.tb45.setCurrentIndex(-1)
            self.l42.setText("")
            self.tabWidget.setCurrentIndex(1)
        except con.Error as e:
            self.l42.setText("Error! Modification Unsuccessful ! ")
            print("Error Occurred in Connecting to school_db " + str(e))
    def delete_attendance_details(self):
        query = QMessageBox.question(
            self, "Delete", "Are you sure you\nWant to delete This Attendance Details?", QMessageBox.StandardButton.Yes |
QMessageBox.StandardButton.No)
        if query == QMessageBox.StandardButton.Yes:
            try:
                mydb = con.connect(host="localhost", user="root",
                                   password="user", db="school_db")
                cursor = mydb.cursor(buffered=True, dictionary=True)
                registration_number = self.cb42.currentText()
                attendance_date = self.cb43.currentText()
                qry = "delete from attendance where registration_number = '" + \
                    registration_number + "' and attendance_date='" + attendance_date + "'"
```

```
            cursor.execute(qry)
            mydb.commit()
            self.l42.setText("Attendance Deleted Successfully !!")
            QMessageBox.information(
                self, "School Management System", "Attendance Deleted Successfully !!")
            self.tb44.setCurrentIndex(-1)
            self.tb45.setCurrentIndex(-1)
            self.l42.setText("")
            self.tabWidget.setCurrentIndex(1)
        except con.Error as e:
            self.l42.setText("Error! Could not Delete Attendance ! ")
            print("Error Occurred in Connecting to school_db " + str(e))
def show_fees(self):
    self.tabWidget.setCurrentIndex(6)
    self.fees_fetch_registration_number()
    self.fill_next_receipt_number()
    self.db51.setDate(QDate.currentDate())
    self.cb53.setCurrentIndex(
        datetime.now().month - 1)
    self.fees_fetch_receipt_number()
def fees_fetch_registration_number(self):
    try:
        self.cb51.clear()
        mydb = con.connect(host="localhost", user="root",
                           password="user", db="school_db")
        cursor = mydb.cursor(buffered=True, dictionary=True)
        cursor.execute("select * from student")
        result = cursor.fetchall()
        if result:
            for stud in result:
                self.cb51.addItem(str(stud['registration_number']))
    except con.Error as e:
        print("Error Occurred in Connecting to school_db " + str(e))
def fill_next_receipt_number(self):
    try:
        rn = 0
        mydb = con.connect(host="localhost", user="root",
                           password="user", db="school_db")
        cursor = mydb.cursor(buffered=True, dictionary=True)
        cursor.execute("select * from fees")
        result = cursor.fetchall()
        if result:
            for stud in result:
                rn += 1
        self.tb51.setText(str(rn+1))
        self.tb51.setReadOnly(True)  # Receipt Number Filed is READ_ONLY
    except con.Error as e:
        print("Error Occurred in Connecting to school_db" + str(e))
def save_fees_details(self):
    try:
        mydb = con.connect(host="localhost", user="root",
                           password="user", db="school_db")
        cursor = mydb.cursor(buffered=True, dictionary=True)
        receipt_number = self.tb51.text()
        registration_number = self.cb51.currentText()
        month = self.cb53.currentText()
        amount = self.tb52.text()
        fees_date = self.db51.date().toString("yyyy-MM-dd")
        qry = "INSERT INTO fees (receipt_number, registration_number, month, amount, fees_date) VALUES (%s, %s, %s, %s, %s)"
        value = (receipt_number, registration_number,
                 month, amount, fees_date)
        cursor.execute(qry, value)
        mydb.commit()
        self.l51.setText("Fees Details Saved Successfully !!")
        self.l51.adjustSize()
        QMessageBox.information(
            self, "School Management System", "Fees Details Saved Successfully !!")
        self.tb52.setText("")
        self.l51.setText("")
        self.l81.setText(str(receipt_number))
        self.l82.setText(str(fees_date))
        self.l84.setText(str(amount))
        self.l85.setText(str(month))
        self.l86.setText(str(registration_number))
        cursor.execute(
            "select * from student where registration_number = '"+registration_number+"'")
        result = cursor.fetchall()
        if result:
            for stud in result:
                self.l83.setText(str(stud['full_name']))
                self.l87.setText(str(stud['phone']))
                self.l88.setText(str(stud['email']))
                self.l89.setText(str(stud['standard']))
        self.tabWidget.setCurrentIndex(8)
```

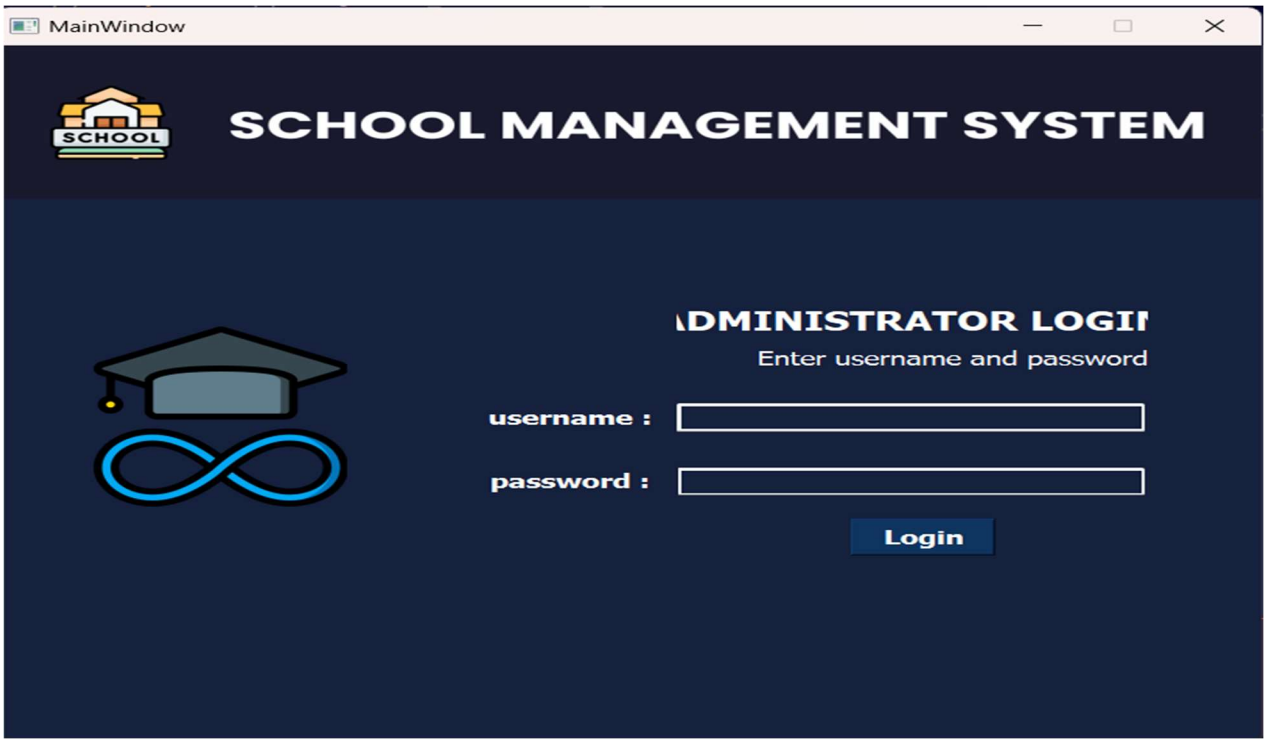
```
        except con.Error as e:
            self.l51.setText("Error! Fees Details Not Saved ! ")
            self.l51.adjustSize()
            print("Error Occurred in Connecting to school_db " + str(e))
    def print_receipt(self):
        printer = QPrinter()
        printer.setResolution(300)
        layout = QPageLayout()
        layout.setOrientation(QPageLayout.Orientation.Landscape)
        layout.setMargins(QMarginsF(0, 0, 0, 0))
        printer.setPageLayout(layout)
        preview = QPrintPreviewDialog(printer, self)
        preview.paintRequested.connect(self.print_preview)
        preview.exec()
    def print_preview(self, printer):
        original_widget = self.tabWidget.currentWidget()
        if original_widget:
            current_widget = QWidget()
            current_widget.resize(original_widget.size())
            current_widget.setStyleSheet("background-color: white;")
            for child in original_widget.findChildren((QLabel, QPushButton)):
                if isinstance(child, QLabel):
                    label = QLabel(current_widget)
                    label.setText(child.text())
                    label.setGeometry(child.geometry())
                    label.setFont(child.font())
                    label.setLayoutDirection(child.layoutDirection())
                    label.setStyleSheet("color: black;")
                if child.objectName() == "name_label":
                    label.setText("karan international")  # Customize the school name here

                if child.objectName() == "address_label":
                    label.setText("chennai")

            pixmap = QPixmap(current_widget.size())
            current_widget.render(pixmap)
            width = printer.width()
            aspect_ratio = pixmap.width() / pixmap.height()
            height = int(width / aspect_ratio)
            scaled_pixmap = pixmap.scaled(
                width, height, Qt.AspectRatioMode.KeepAspectRatio)
            painter = QPainter(printer)
            painter.drawPixmap(0, 0, scaled_pixmap)
            painter.end()
    def fees_fetch_receipt_number(self):
        try:
            self.cb52.clear()
            mydb = con.connect(host="localhost", user="root",
                               password="user", db="school_db")
            cursor = mydb.cursor(buffered=True, dictionary=True)
            cursor.execute("select * from fees")
            result = cursor.fetchall()
            if result:
                for stud in result:
                    self.cb52.addItem(str(stud['receipt_number']))
        except con.Error as e:
            print("Error Occurred in Connecting to school_db " + str(e))
    def fetch_receipt_details(self):
        try:
            receipt_number = self.cb52.currentText()
            mydb = con.connect(host="localhost", user="root",
                               password="user", db="school_db")
            cursor = mydb.cursor(buffered=True, dictionary=True)
            cursor.execute(
                "select * from fees where receipt_number = '" + receipt_number + "'")
            result = cursor.fetchall()
            if result:
                for stud in result:
                    self.tb53.setText(str(stud['registration_number']))
                    self.tb53.setReadOnly(True)
                    self.cb54.setCurrentText(str(stud['month']))
                    self.tb54.setText(str(stud['amount']))
                    self.db52.setDate(QDate.fromString(
                        str(stud['fees_date']), "dd-MM-yyyy"))
        except con.Error as e:
            print("Error Occurred in Connecting to school_db " + str(e))
    def update_fees_details(self):
        try:
            mydb = con.connect(host="localhost", user="root",
                               password="user", db="school_db")
            cursor = mydb.cursor(buffered=True, dictionary=True)
            registration_number = self.tb53.text()
            receipt_number = self.tb52.text()
            month = self.cb54.currentText()
```

```
            amount = self.tb54.text()
            fees_date = self.db52.text()
            qry = "UPDATE fees SET month = %s, amount = %s, fees_date = %s WHERE receipt_number = %s"
            values = (month, amount, fees_date, receipt_number)
            cursor.execute(qry, values)
            mydb.commit()
            self.l52.setText("Fees Details Modified Successfully !!")
            self.l52.adjustSize()
            QMessageBox.information(
                self, "School Management System", "Fees Details Modified Successfully !!")
            self.tb53.setText("")
            self.tb54.setText("")
            self.l52.setText("")
            self.l81.setText(str(receipt_number))
            self.l82.setText(str(fees_date))
            self.l84.setText(str(amount))
            self.l85.setText(str(month))
            self.l86.setText(str(registration_number))
            cursor.execute(
                "select * from student where registration_number = '"+registration_number+"'")
            result = cursor.fetchall()
            if result:
                for stud in result:
                    self.l83.setText(str(stud['full_name']))
                    self.l87.setText(str(stud['phone']))
                    self.l88.setText(str(stud['email']))
                    self.l89.setText(str(stud['standard']))
            self.tabWidget.setCurrentIndex(8)
        except con.Error as e:
            self.l52.setText("Error! Fees Details Not Modified ! ")
            self.l52.adjustSize()
            print("Error Occurred in Connecting to school_db " + str(e))
    def delete_fees_details(self):
        query = QMessageBox.question(
            self, "Delete", "Are you sure you\nWant to delete This Fees Details?", QMessageBox.StandardButton.Yes |
QMessageBox.StandardButton.No)
        if query == QMessageBox.StandardButton.Yes:
            try:
                mydb = con.connect(host="localhost", user="root",
                                   password="user", db="school_db")
                cursor = mydb.cursor(buffered=True, dictionary=True)
                receipt_number = self.cb52.currentText()
                qry = "delete from fees where receipt_number = '" + \
                    receipt_number + "'"
                cursor.execute(qry)
                mydb.commit()
                self.l52.setText("Fees Details Deleted Successfully !!")
                self.l52.adjustSize()
                QMessageBox.information(
                    self, "School Management System", "Fees Details Deleted Successfully !!")
                self.tb53.setText("")
                self.tb54.setText("")
                self.cb54.setCurrentIndex(0)
                self.l52.setText("")
                self.tabWidget.setCurrentIndex(1)
            except con.Error as e:
                self.l52.setText("Error! Could not Delete Fees Details ! ")
                self.l52.adjustSize()
                print("Error Occurred in Connecting to school_db " + str(e))
    def show_report(self):
        table_name = self.sender()
        self.l61.setText(table_name.text())  # Set the table Name
        self.tabWidget.setCurrentIndex(7)
        try:
            self.tableReport.setRowCount(0)
            if (table_name.text() == "Students Reports"):
                mydb = con.connect(host="localhost", user="root",
                                   password="user", db="school_db")
                cursor = mydb.cursor(buffered=True, dictionary=True)
                qry = "select registration_number, full_name, gender, dob, age, address, phone, email, standard from student"
                cursor.execute(qry)
                result = cursor.fetchall()
                self.tableReport.clear()
                self.tableReport.setRowCount(len(result))
                self.tableReport.setColumnCount(len(result[0]))
                for row_number, row_data in enumerate(result):
                    for column_number, data in enumerate(row_data.values()):
                        self.tableReport.setItem(
                            row_number, column_number, QTableWidgetItem(str(data)))
                self.tableReport.setHorizontalHeaderLabels(
                    ['REG NO', 'NAME', 'GENDER', 'DOB', 'AGE', 'ADDRESS', 'PHONE NO', 'EMAIL ID', 'STANDARD'])
                self.tableReport.resizeColumnsToContents()
                self.tableReport.resizeRowsToContents()
            if (table_name.text() == "Marks Reports"):
```

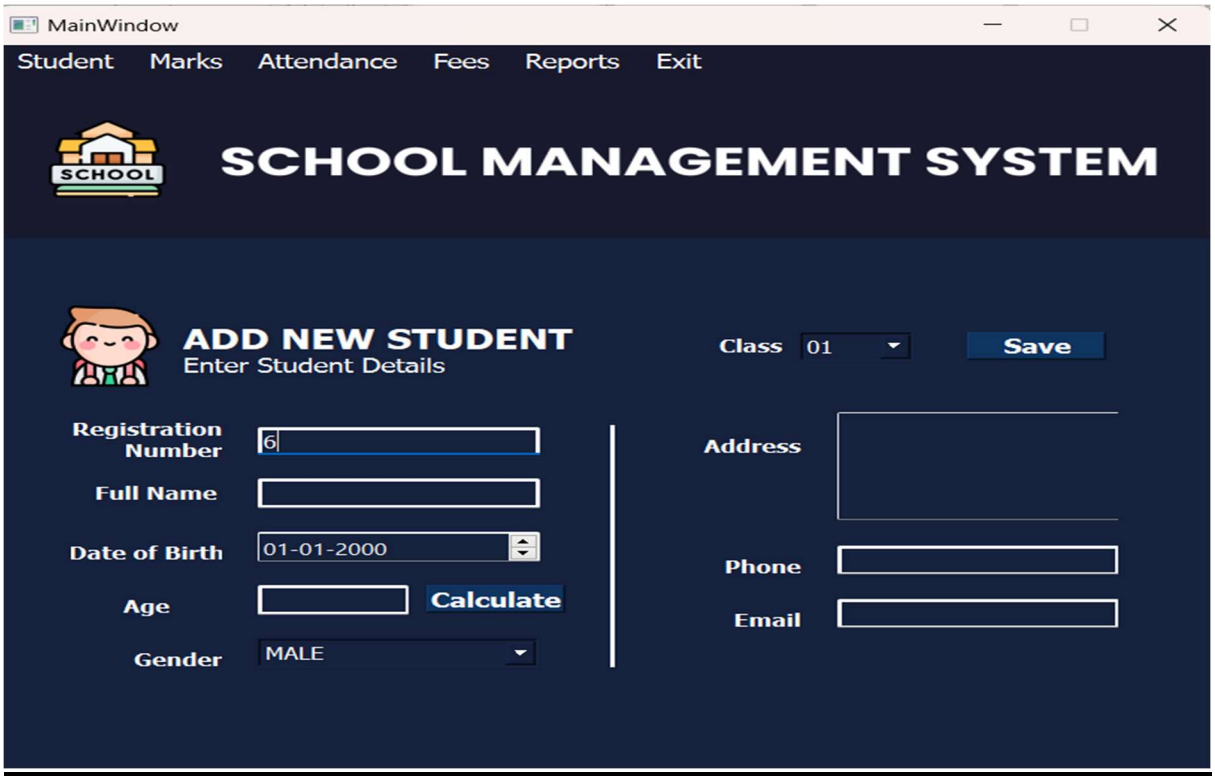
```
                mydb = con.connect(host="localhost", user="root",
                                   password="user", db="school_db")
                cursor = mydb.cursor(buffered=True, dictionary=True)
                qry = "select registration_number, exam_name, language, maths, science, arts from marks"
                cursor.execute(qry)
                result = cursor.fetchall()
                self.tableReport.clear()
                self.tableReport.setRowCount(len(result))
                self.tableReport.setColumnCount(len(result[0]))
                for row_number, row_data in enumerate(result):
                    for column_number, data in enumerate(row_data.values()):
                        self.tableReport.setItem(
                            row_number, column_number, QTableWidgetItem(str(data)))
                self.tableReport.setHorizontalHeaderLabels(
                    ['REG NO', 'EXAM NAME', 'LANGUAGE', 'MATHS', 'SCIENCE', 'ARTS'])
                self.tableReport.resizeColumnsToContents()
                self.tableReport.resizeRowsToContents()
            if (table_name.text() == "Attendance Reports"):
                mydb = con.connect(host="localhost", user="root",
                                   password="user", db="school_db")
                cursor = mydb.cursor(buffered=True, dictionary=True)
                qry = "select registration_number, attendance_date, morning, evening from attendance"
                cursor.execute(qry)
                result = cursor.fetchall()
                self.tableReport.clear()
                self.tableReport.setRowCount(len(result))
                self.tableReport.setColumnCount(len(result[0]))
                for row_number, row_data in enumerate(result):
                    for column_number, data in enumerate(row_data.values()):
                        self.tableReport.setItem(
                            row_number, column_number, QTableWidgetItem(str(data)))
                self.tableReport.setHorizontalHeaderLabels(
                    ['REG NO', 'ATTENDANCE DATE', 'MORNING', 'EVENING'])
                self.tableReport.resizeColumnsToContents()
                self.tableReport.resizeRowsToContents()
            if (table_name.text() == "Fees Reports"):
                mydb = con.connect(host="localhost", user="root",
                                   password="user", db="school_db")
                cursor = mydb.cursor(buffered=True, dictionary=True)
                qry = "select receipt_number, registration_number, month, amount, fees_date from fees"
                cursor.execute(qry)
                result = cursor.fetchall()
                self.tableReport.clear()
                self.tableReport.setRowCount(len(result))
                self.tableReport.setColumnCount(len(result[0]))
                for row_number, row_data in enumerate(result):
                    for column_number, data in enumerate(row_data.values()):
                        self.tableReport.setItem(
                            row_number, column_number, QTableWidgetItem(str(data)))
                self.tableReport.setHorizontalHeaderLabels(
                    ['RECEIPT NO', 'REG NO', 'MONTH', 'AMOUNT', 'FEES DATE'])
                self.tableReport.resizeColumnsToContents()
                self.tableReport.resizeRowsToContents()
        except con.Error as e:
            print("Error Occurred in Connecting to school_db " + str(e))
    def logout(self):
        self.menubar.setVisible(False)
        self.tb01.setText("")
        self.tb02.setText("")
        self.l01.setText("")
        self.tabWidget.setCurrentIndex(0)
        QMessageBox.information(
            self, "School Management System", "Logged Out Successfully !")
def main():
    app = QApplication(sys.argv)
    window = MainApp()
    window.show()
    app.exec()
if __name__ == '__main__':
    main()
```

EXPERIMENT RESULTS AND ANALYSIS

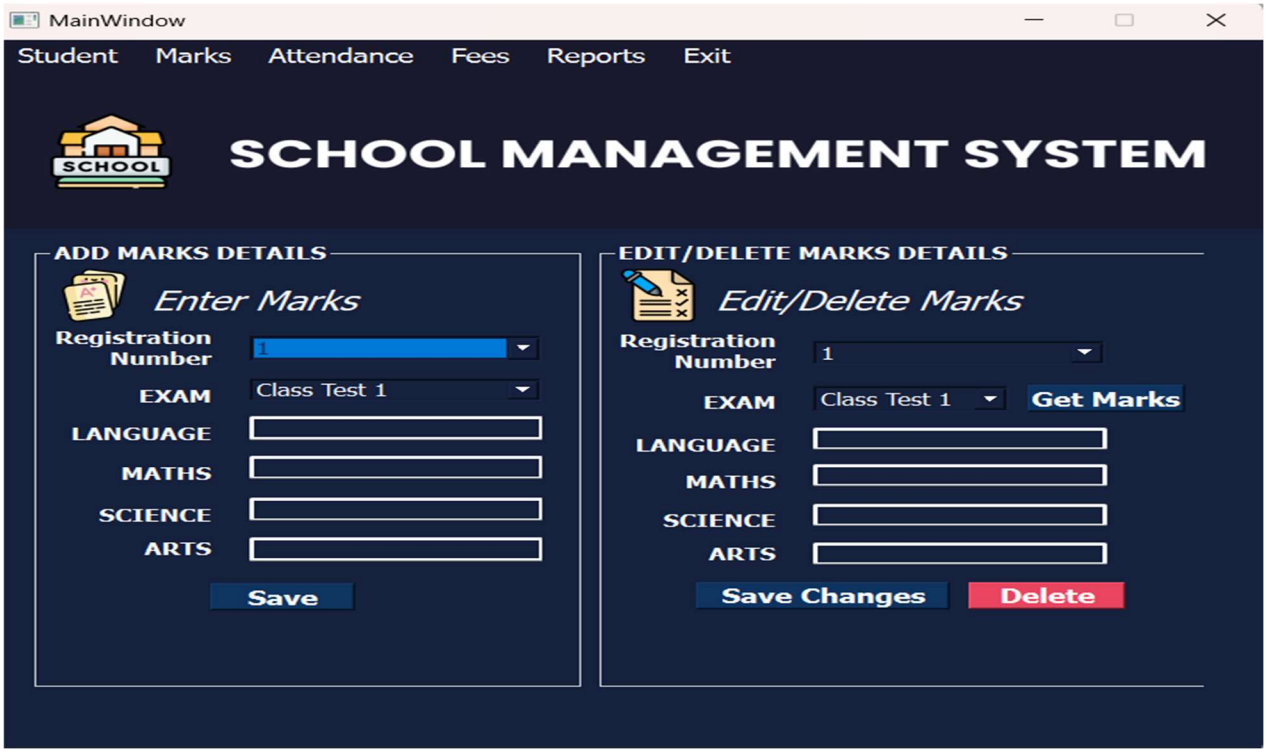
HOME PAGE



STUDENT DETAILS PAGE

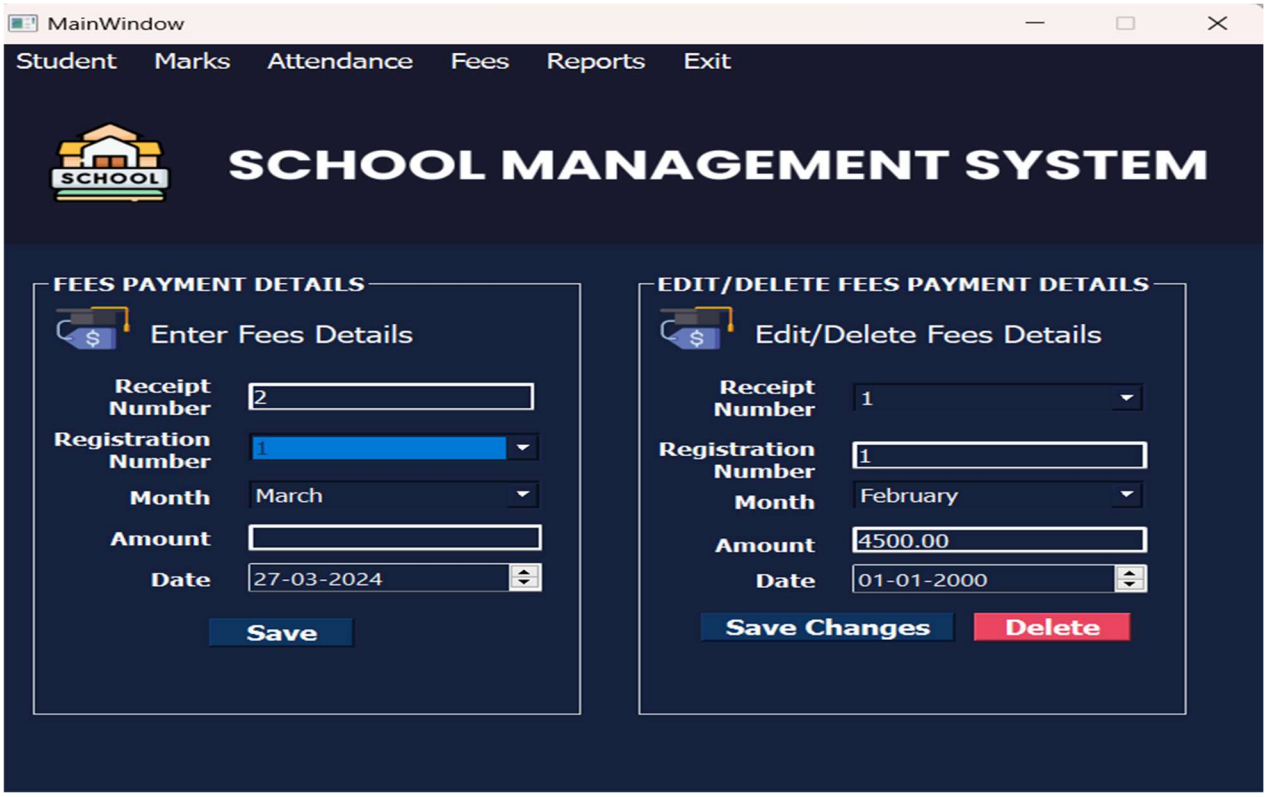


MARKS PAGE

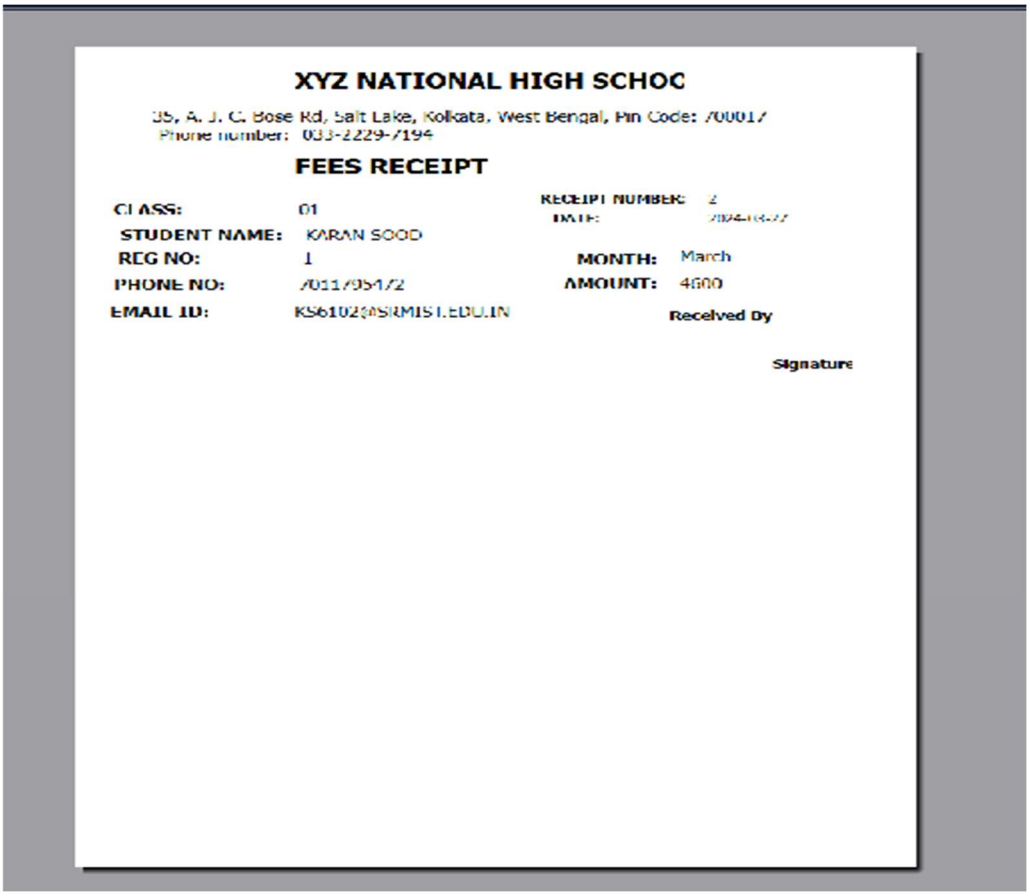


ATTENDANCE PAGE

FEES PAGE



FEES RECEIPT



XYZ NATIONAL HIGH SCHOC

35, A. J. C. Bose Rd, Salt Lake, Kolkata, West Bengal, Pin Code: 700017
Phone number: 033-2229-7194

FEES RECEIPT

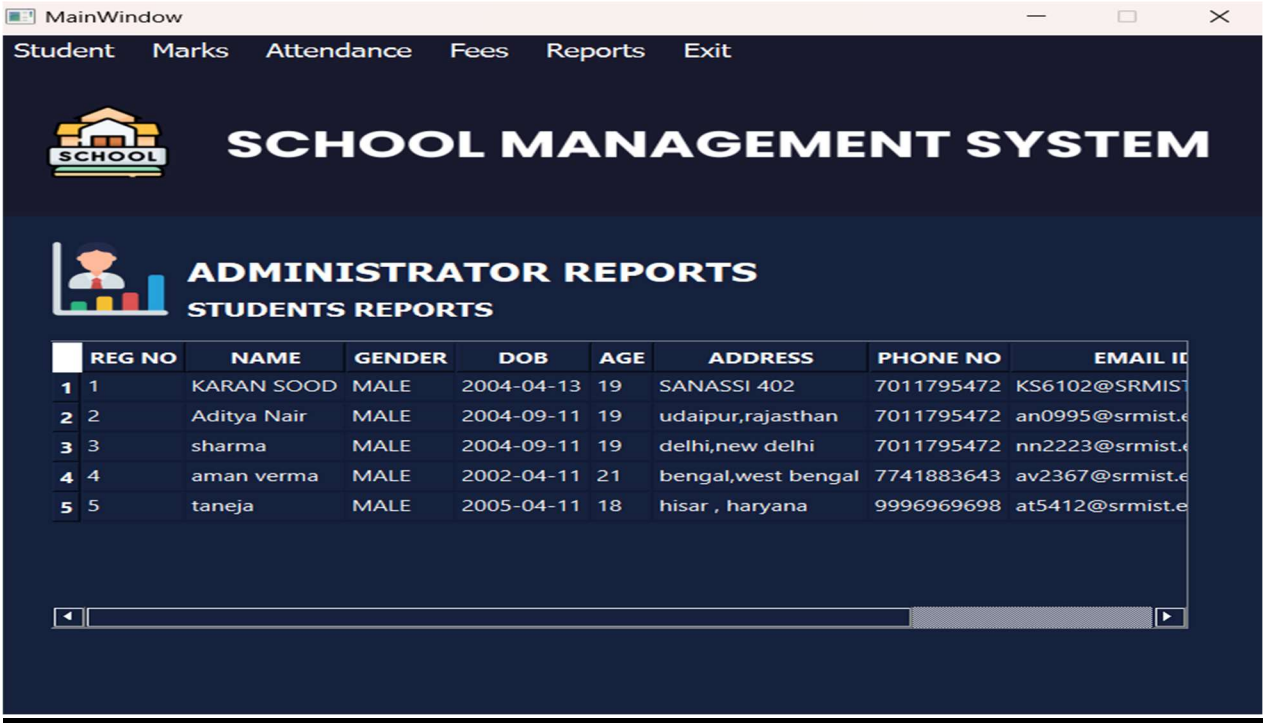
CLASS: 01
STUDENT NAME: KARAN SOOD
REG NO: 1
PHONE NO: 7011795472
EMAIL ID: KS6102@SRMIST.EDU.IN

RECEIPT NUMBER: 2
DATE:
MONTH: March
AMOUNT: 4600

Received By

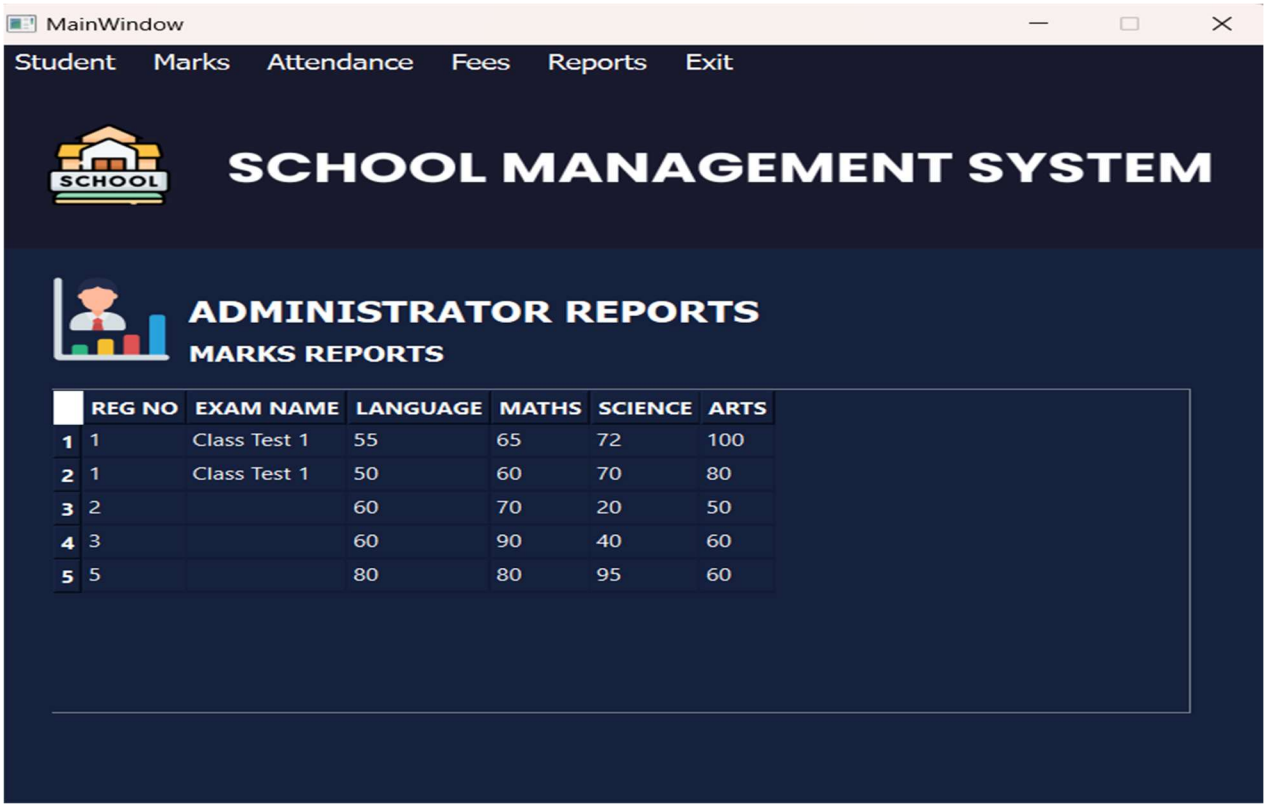
Signature

STUDENT REPORT PAGE



	REG NO	NAME	GENDER	DOB	AGE	ADDRESS	PHONE NO	EMAIL ID
1	1	KARAN SOOD	MALE	2004-04-13	19	SANASSI 402	7011795472	KS6102@SRMIST
2	2	Aditya Nair	MALE	2004-09-11	19	udaipur,rajasthan	7011795472	an0995@srmist.e
3	3	sharma	MALE	2004-09-11	19	delhi,new delhi	7011795472	nn2223@srmist.e
4	4	aman verma	MALE	2002-04-11	21	bengal,west bengal	7741883643	av2367@srmist.e
5	5	taneja	MALE	2005-04-11	18	hisar , haryana	9996969698	at5412@srmist.e

MARKS REPORT

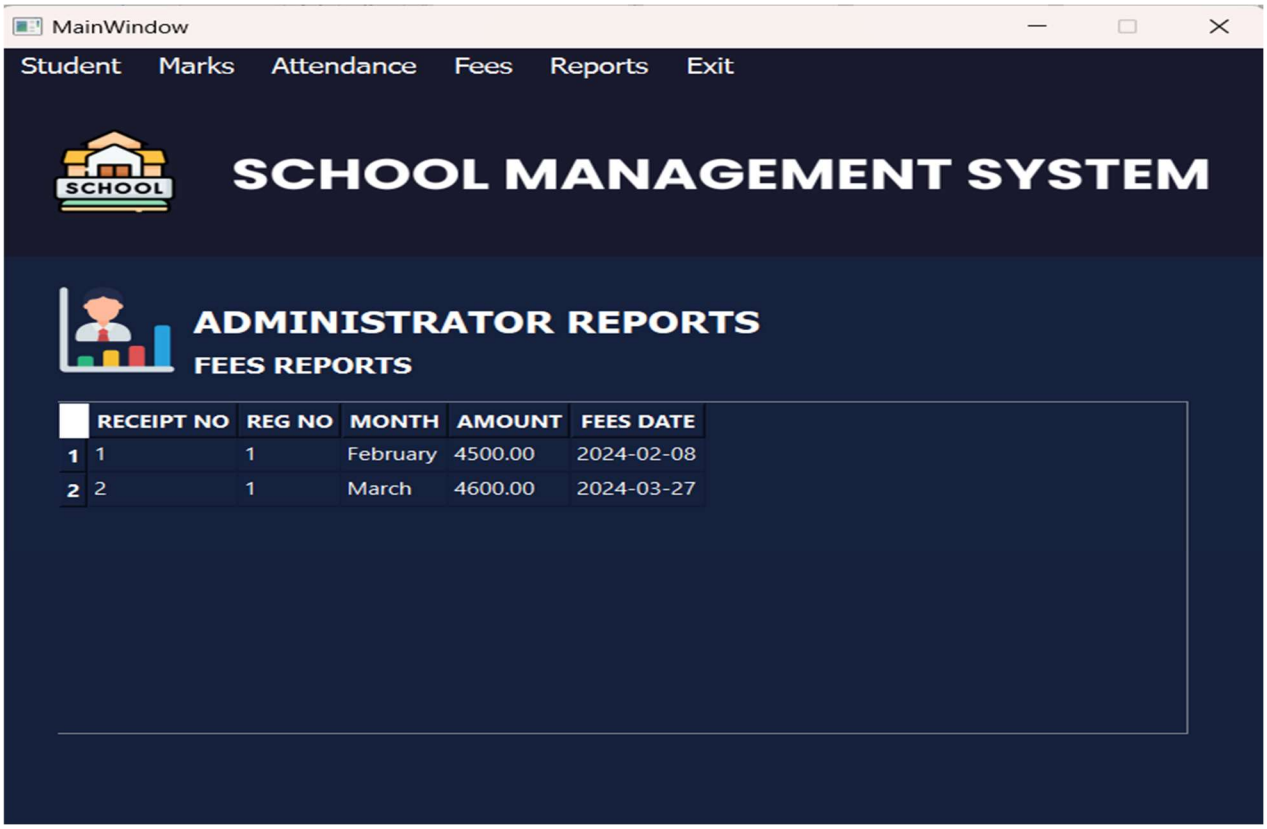


	REG NO	EXAM NAME	LANGUAGE	MATHS	SCIENCE	ARTS
1	1	Class Test 1	55	65	72	100
2	1	Class Test 1	50	60	70	80
3	2		60	70	20	50
4	3		60	90	40	60
5	5		80	80	95	60

ATTENDANCE REPORT

	REG NO	ATTENDANCE DATE	MORNING	EVENING
1	1	2024-02-08	ABSENT	ABSENT

FEES REPORT



	RECEIPT NO	REG NO	MONTH	AMOUNT	FEES DATE
1	1	1	February	4500.00	2024-02-08
2	2	1	March	4600.00	2024-03-27

CONCLUSION

In conclusion, the school management system presented offers a comprehensive solution for efficiently managing attendance records within a school environment. By leveraging a relational database management system, such as MySQL, and integrating it with a user-friendly graphical interface, the system provides administrators with the tools needed to effortlessly track and update attendance data for students. Through features like fetching registration numbers, retrieving attendance dates, saving, updating, and deleting attendance details, the system streamlines administrative tasks, saving time and effort. Furthermore, the system prioritizes data integrity and security by utilizing parameterized queries to prevent SQL injection vulnerabilities and ensuring proper handling of database connections. Overall, the school management system represents a reliable and effective solution for enhancing attendance management processes in educational institutions.

REFERENCE

1. Geeks For Geeks
   https://www.geeksforgeeks.org

2. ChatGPT
   https://chat.openai.com/

3. Coding Ninjas
   https://www.codingninjas.com/landing/cnsat/?utm_source=google&utm_medium=[search]&utm_campaign=20131922872_149942135238_e_coding%20ninjas__658412799015_m_____9300909&gad_source=1&gclid=Cj0KCQjw4vKpBhCZARIsAOKHoWSpFlBO_5rgdFtxoMToX9WlF_bG-QwVxjyCt8PqUq_076jV0dUOnY0aAtEEEALw_wcB

4. Quora
   https://www.quora.com