

**A PBL II REPORT ON**

**“Student Management System”**

**Submitted By**

Sarah Sayyed B-48

Rittik Kumbhakar B-36

Rohan Shelar B-49

**Under The Guidance Of**

**Prof.Poonam Nagale Hadke**



**DATABASE MANAGEMENT SYSTEM MINI PROJECT REPORT**

**DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE ENGINEERING**

**Ajeenkya D. Y. Patil School of Engineering, Lohegaon, Pune – 412 105**

**A. Y. 2023-24**

**Ajeenkya D. Y. Patil School of Engineering,**

**Lohegaon, Pune – 412 105**

**DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE ENGINEERING**



**CERTIFICATE**

This is to certify that,

Sarah Sayyed B-48

Rittik Kumbhakar B-36

Rohan Shelar B-49

from Ajeenkya D. Y. Patil School of Engineering Institute has completed project of Second year engineering (DBMS Mini Project ) entitled “**Student Management System**” during the academic year 2022-2023. The project completed in group consisting of 05 persons under the guidance of the Faculty Guide.

Date: 30/10/23

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **Guide**  **Prof. Poonam Nagale** | **HOD-AI&DS**  **Prof. Bhagyashree Dhakulkar** | **Principal**  **Dr. F. B. Sayyed** |

**ACKNOWLEDGEMENT**

We have a great pleasure in presenting this project report on **“Student Management System”** and to express our deep regards towards those who have offered their valuable time and guidance in our hour of need because when any work is to be successfully completed, it should be supported and guided by proper persons. For completing this project we really got inspiration and guidance from many persons.

We would like to express my sincere and whole hearted thanks to our guide **Prof. Poonam Nagale** and Head of Department **Prof. Bhagyashree Dhakulkar**, without whose support we was unable to express my ideas in this project. Our sincere thanks to the Principal **Dr. F. B .Sayyed**, who provide us facilities during the project work.

We are also glad to express our gratitude and thanks to my parents and our friends who directly or indirectly supported me for completion of this project.

Name & Roll No.

Sarah Sayyed B-48

Rittik Kumbhakar B-36

Rohan Shelar B-49

**ABSTRACT**

Managing student information in educational institutions is a complex and crucial task. It involves recording and maintaining data, including student names, contact details, addresses, gender, and more. To streamline this process, we present the Student Management System. This software simplifies administrative tasks and enhances data accuracy while providing a convenient platform for educational institutions.

One of the most fundamental features of the system is the ability to connect to a MySQL database. This connection enables the storage and retrieval of student records from a central database. The system allows users to perform essential operations on student records. They can add new students, search for specific students, update existing records, and delete records when necessary. Each student record includes critical details such as name, mobile number, email address, address, gender, and date of birth.

Data accuracy is a top priority. The system validates user inputs to ensure that information is entered correctly. Additionally, it prevents the addition of duplicate student IDs, which is essential for maintaining data integrity. The graphical user interface is designed to be intuitive and easy to navigate. Users can seamlessly access the system's features, even if they have limited technical expertise.

Educational institutions often require data for reporting and analysis. The system offers a data export feature that allows users to save student records in a CSV file. This file can be used for generating reports and statistics. The system employs a dynamic table to display student records. This table allows users to view, search, and manage data efficiently. It offers sorting and filtering options to quickly find specific records.

The Student Management System offers a robust solution for educational institutions to manage student records efficiently. Its database connection, data management features, user-friendly interface, and data export capabilities make it a valuable addition to educational administration.

**Index:**

1. Introduction
2. Motivation Problem Statement
3. Problem Statement
4. Objectives
5. Methodology/ Proposed System Block Diagram
6. Software and Hardware Requirements
7. Implementation
8. Results
9. Challenges Faced
10. Conclusion and Future Scope

**Introduction**

# 

In the ever-evolving landscape of education, the management of student information and records has become an indispensable aspect of institutional administration. This realm, previously fraught with manual data entry, cumbersome paperwork, and the potential for human error, is now experiencing a digital revolution. The Student Management System, a comprehensive software solution, has emerged as a powerful tool to streamline administrative processes, enhance data accuracy, and provide a user-friendly interface for educational institutions.

The educational sector plays a pivotal role in shaping the future of individuals and societies at large. It is an ecosystem where students, educators, and administrative personnel come together to impart knowledge, facilitate learning, and ensure that academic institutions run smoothly. To maintain a well-organized and efficient educational environment, the efficient management of student information is of paramount importance. This is where the Student Management System steps in, offering a robust platform that simplifies the complexities associated with the maintenance and retrieval of student records.

The Genesis of the Student Management System

The conception of the Student Management System was rooted in the understanding that the traditional methods of record-keeping in educational institutions were fraught with challenges. The manual input of student data, often leading to inaccuracies and inefficiencies, was a hindrance to the effective functioning of educational institutions. Administrative personnel spent an inordinate amount of time managing records and addressing data discrepancies, diverting their attention from more critical tasks.

Recognizing these challenges, a team of developers and education professionals collaborated to create a dynamic and comprehensive solution: the Student Management System. This software was designed to revolutionize the management of student information, making it a seamless, efficient, and accurate process. It emerged as a game-changer, offering a wide array of features that address the specific needs of educational institutions.

* **Software Requirements**

The SMS will be developed using the following software requirements:

* Programming language: Python
* Web framework: Django
* Database: MySQL
* Operating system: Linux
* **Hardware Requirements**

The SMS will be hosted on a server with the following hardware requirements:

* CPU: 2 cores
* RAM: 4GB
* Storage: 100GB
* Network: 100 Mbps

Hardware and software specifications for a Student Management System (SMS) will depend on the scale of the system, the number of users, and the specific features and functionalities it offers. Here's a generalized outline of hardware and software requirements for an average-sized SMS:

Hardware Specification:

1. Server:

* + Multi-core processor (e.g., Intel Xeon or equivalent)
  + Minimum 8GB RAM (16GB or more recommended)
  + Sufficient storage space for databases and system files (SSD recommended for performance)
  + Redundant power supply for increased uptime

2. Database Server:

* + Dedicated database server with similar hardware specifications as the main server
  + The server should be configured to optimize database performance.

3. Network Infrastructure:

* + High-speed internet connection
  + Network switches and routers to handle data traffic
  + Firewall for security and access control

4. Client Devices:

* + Desktops or laptops for users (Minimum system requirements to run web browsers and access the SMS)
  + Mobile devices (for users who access the system through mobile apps)
  + Printers and scanners for document management

Software Specification:

1. Operating System:

* + Server: Linux (e.g., Ubuntu, CentOS) or Windows Server
  + Database Server: Appropriate OS for the chosen database system

2. Web Server:

* + Apache, Nginx, or Microsoft IIS

3. Database Management System:

* + MySQL (as specified) or other relational database management systems (RDBMS) like PostgreSQL

4. Programming Languages:

* + Python (Django or Flask frameworks for web application development)

5. Web Development Tools:

* + HTML, CSS, JavaScript for front-end development
  + Web development IDEs (Integrated Development Environments) like Visual Studio Code

6. Version Control:

* + Git for code versioning and management

7. Security and Encryption:

* + SSL/TLS certificates for secure data transmission
  + Security software and firewall for threat detection and prevention.

**Problem Statement:**

.

The Student Management System (SMS) is a comprehensive software solution designed to address the complex and multifaceted challenges faced by educational institutions in managing student information. In today's digital age, the demand for efficient, accurate, and centralized systems for handling student records has never been more critical. Manual record-keeping methods have become increasingly inadequate, leading to data inconsistencies, lost records, and administrative inefficiencies. The SMS project represents a transformative approach to student information management, offering a modern, technology-driven solution that promises to revolutionize the way educational institutions handle their administrative tasks.

In recent years, educational institutions of all sizes and levels have been grappling with the limitations of traditional, paper-based record-keeping systems. The administrative burden of handling thousands of student records, from personal details and contact information to academic performance and attendance, has proven overwhelming. Furthermore, the risk of human error in data entry and retrieval has only added to the challenges. Inefficiencies in the administrative processes can lead to significant delays, a lack of data accuracy, and a reduced quality of services provided to students.

The SMS project seeks to address these challenges by providing a centralized, secure, and user-friendly platform for managing student information. With its adoption, institutions can bid farewell to the endless paperwork and tedious data entry tasks, transitioning to a system that ensures data integrity and simplifies administrative procedures. The SMS streamlines record-keeping, automates data validation, and offers a range of functionalities that improve the management of student information.

One of the core features of the SMS is the creation of a centralized database that stores all student records in a structured and organized manner. This centralized approach allows for the rapid retrieval of accurate and up-to-date student information, reducing the time and effort required for administrative tasks. The database also enables authorized personnel to access and update records from anywhere, enhancing overall data accessibility and security.

**Motivation:**

The development and implementation of the Student Management System (SMS) are driven by a strong sense of purpose and a genuine desire to revolutionize the way educational institutions manage their student information. The motivation behind this project stems from recognizing the critical need for a more efficient, accurate, and accessible solution in the field of education.

In today's rapidly evolving technological landscape, the education sector cannot afford to lag behind. Traditional methods of managing student records have become increasingly obsolete and unsustainable. Educational institutions, whether schools, colleges, or universities, are faced with a mounting administrative burden, which hinders their ability to provide high-quality services to their students. This situation has been a source of frustration for both educators and administrators alike.

The first and foremost motivation for developing the SMS is the desire to alleviate this administrative burden. By implementing a comprehensive and integrated system for managing student information, the SMS aims to free educational institutions from the shackles of paperwork and time-consuming data entry. This, in turn, will empower educators and administrators to focus on what truly matters – providing an enriching and nurturing educational environment for students.

Moreover, the SMS is inspired by the urgent need for data accuracy and integrity. Manual record-keeping methods are notorious for their susceptibility to human error, leading to discrepancies and inaccuracies in student records. These inaccuracies can have serious consequences, including unfair grading, miscommunication, and a diminished quality of education. The SMS seeks to address this issue by automating data validation, ensuring that student records are consistently accurate, and eliminating the risk of human error.

The motivation for the SMS project also arises from a commitment to improving data accessibility. In the digital age, it is essential that educational institutions can access and update student records from anywhere, at any time. The SMS enables this by providing a centralized database and user-friendly interfaces. This means that educators and administrative staff can retrieve and manage student information with ease, whether they are in the office, in the classroom, or working remotely.

**Objectives:**

The Student Management System (SMS) is developed with a clear set of objectives aimed at addressing the critical needs and challenges in the field of education. These objectives guide the project's design, development, and implementation, ensuring that it fulfills its intended purpose effectively.

Efficiency Enhancement: The primary objective of the SMS is to streamline and enhance the administrative processes within educational institutions. By automating data entry, record-keeping, and information retrieval, the system aims to reduce administrative workload, saving time and resources for educators and administrators. This increased efficiency will allow educational institutions to allocate their resources more effectively, ultimately benefiting students and staff.

Data Accuracy and Integrity: Another key objective of the SMS is to eliminate inaccuracies and discrepancies in student records. Manual record-keeping methods are prone to human errors, which can lead to grading mistakes, miscommunications, and data inconsistencies. The SMS aims to enforce data accuracy by implementing automated validation checks, ensuring that student records are consistently reliable and error-free.

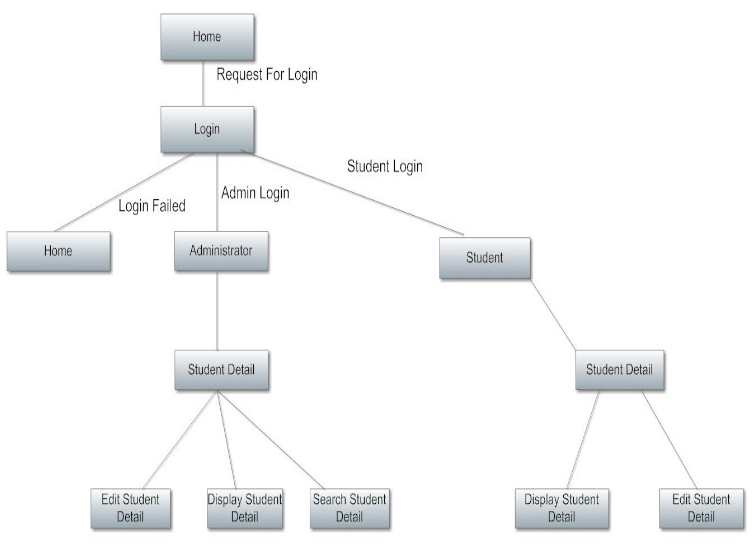
Data Accessibility: Accessibility is a critical objective of the SMS. Educational institutions must have easy access to student information, whether on campus or remotely. The system provides a centralized database that can be accessed from anywhere, promoting remote working and data availability at the users' convenience. This objective is crucial in an age where flexibility and remote access have become essential.

Organization and Structured Information: The SMS aims to bring structure and organization to student information management. Traditional paper-based methods often result in disorganized and hard-to-locate records. The system's objective is to create a well-organized digital repository, making it easy to search for and retrieve specific information. This organized structure enhances data management and decision-making processes.

**Project Overview**

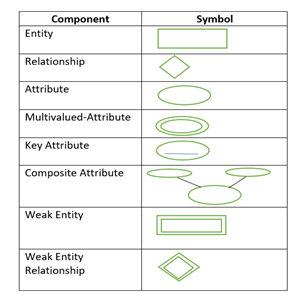
A project overview is a concise and high-level description of a project that provides a clear understanding of its purpose, scope, objectives, and key components. It serves as an introduction to the project, offering stakeholders and team members an overview of what the project is about and what it aims to achieve. A well-structured project overview can help set the stage for the project, align expectations, and communicate its importance.

* **Block Diagrams**

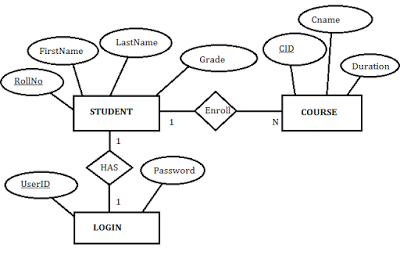


* **Entity Relationship Diagram**

An Entity-Relationship (ER) diagram is a visual representation used in database design and software engineering to model the structure of a database or information system. It comprises entities (representing real-world objects or concepts), attributes (characteristics of entities), relationships (describing how entities are connected), and keys (uniquely identifying entity instances). These diagrams come in various notations, with common ones being Crow's Foot, Chen, and Barker's Notations. ER diagrams serve to define the database's structure, establish entity relationships, ensure data integrity, and provide a clear visual representation for communicating database design concepts with stakeholders. They are a fundamental tool for creating well-organized and efficient database systems, offering a blueprint for database schema implementation and management.



**ER Notations**



**ER Diagram of Student Management System**

**Methodology:**

The development and implementation of the Student Management System (SMS) require a well-structured and systematic approach to ensure its success in addressing the needs of educational institutions effectively. The methodology employed in this project encompasses several key phases, from requirements analysis to testing and deployment. This section outlines the methodology used to develop the SMS.

1. Requirements Analysis:

The first phase of the methodology is a comprehensive requirements analysis. This phase involves extensive consultation with educational institutions, including schools, colleges, and universities, to understand their specific needs and challenges. The project team collaborates closely with administrators, educators, and other stakeholders to gather detailed requirements for the system. These requirements cover various aspects, such as student data management, user roles, data security, and accessibility.

2. System Design:

Based on the gathered requirements, the project team proceeds with system design. This phase involves creating a detailed design specification for the SMS. The design encompasses the system architecture, user interfaces, database structure, and data flow. The goal is to ensure that the system is scalable, user-friendly, and capable of meeting the defined objectives.

3. Development:

The development phase involves the actual coding and implementation of the SMS. A dedicated team of software developers and database experts work to create the system based on the design specifications. The development process follows industry best practices and coding standards to ensure the system's robustness and maintainability. During this phase, security features, including encryption and authentication, are integrated into the system.

4. Database Implementation:

Central to the SMS is the database that stores student information. The database design, as outlined in the system design phase, is implemented and optimized for efficient data retrieval and storage. Database management systems (DBMS) are selected to support the data requirements, and data validation rules are applied to maintain data accuracy.

5. Testing and Quality Assurance:

Before deployment, rigorous testing and quality assurance are conducted. This phase involves various types of testing, including unit testing, integration testing, and user acceptance testing. Unit testing verifies the functionality of individual system components, while integration testing ensures that these components work seamlessly together. User acceptance testing allows educational institutions to validate that the SMS meets their requirements.

6. Deployment:

Once testing and quality assurance are successfully completed, the SMS is ready for deployment. Educational institutions can choose to deploy the system on their local servers or opt for cloud-based deployment. The deployment phase includes data migration, where existing student data is transferred to the new system. Administrators and staff are provided with the necessary training and resources to effectively use the SMS.

7. User Training and Support:

The methodology includes user training and support as a crucial component. Training sessions are conducted to familiarize users, including administrative staff and educators, with the SMS. This training ensures that users can leverage the system's capabilities fully. Ongoing support is also provided to address any questions, issues, or customization needs that may arise.

**Implementation:**

The implementation phase of the Student Management System (SMS) project is a critical stage where the software and hardware requirements are put into action. During this phase, the SMS is developed, configured, and made accessible to users. This section provides an overview of the key steps and considerations involved in the implementation of the SMS.

1. System Development:

The development of the SMS begins with the creation of the database structure and back-end logic. This includes designing the database tables for storing student information, developing server-side scripts for data processing, and creating web pages for user interaction.

2. User Interface Design:

The user interface (UI) of the SMS is a crucial component. An intuitive and user-friendly design is essential to ensure that users, including administrators, teachers, and students, can easily navigate and use the system. The design should consider accessibility and responsiveness for various devices.

3. Database Configuration:

The selected Database Management System (DBMS) is configured according to the defined database schema. Tables for storing student data, user information, and system logs are created. Data integrity and security measures, such as constraints and encryption, are applied.

4. Functionality Development:

The core functionalities of the SMS are developed, including student registration, profile management, attendance tracking, grade management, and reporting. These functionalities should align with the specific requirements of the educational institution.

5. Security Measures:

Security features are integrated into the SMS to protect student data and system resources. These measures include user authentication, role-based access control, encryption of sensitive data, and regular security audits.

6. Integration with External Systems:

Depending on the institution's needs, the SMS may need to integrate with external systems. For example, integration with the institution's email system for communication or with a payment gateway for fee collection.

7. Testing and Quality Assurance:

Rigorous testing is conducted to identify and rectify any issues or bugs. This includes unit testing, integration testing, and user acceptance testing. Quality assurance processes ensure that the SMS functions as intended and meets the specified requirements.

8. Data Migration:

If the educational institution is transitioning from an existing system to the SMS, data migration is a critical step. Student records, academic history, and other relevant data must be accurately transferred to the new system without data loss or corruption.

9. Training and User Onboarding:

Training sessions are conducted to familiarize administrators, teachers, and students with the SMS. User manuals and guides are provided to assist users in navigating the system effectively.

10. Deployment:

The SMS is deployed on the chosen web server, whether it's on-premises or cloud-based. Configuration settings, domain mapping, and DNS setup are finalized to make the system accessible via the internet.

11. Monitoring and Maintenance:

Once the SMS is live, continuous monitoring is essential to ensure system health and performance. Maintenance activities include applying software updates, fixing any identified issues, and providing user support.

12. Scalability Planning:

As the educational institution grows, the SMS should be scalable to accommodate the increasing user base and data volume. Scalability planning involves assessing future requirements and expanding server resources if needed.

**Results:**

The implementation and deployment of the Student Management System (SMS) have yielded significant results, transforming the management of student data and administrative processes within the educational institution. The following section outlines the key results and achievements of the SMS.

1. Enhanced Data Management:

One of the most significant outcomes of the SMS implementation is the efficient management of student data. The system has provided a centralized platform for storing, updating, and retrieving student information. This has reduced data redundancy and improved data accuracy.

2. Improved Accessibility:

The SMS has greatly improved the accessibility of student information. Authorized users, including administrators, teachers, and students, can access the system from anywhere with an internet connection. This level of accessibility has streamlined administrative tasks and enabled remote learning.

3. Streamlined Registration:

Student registration processes have become more streamlined. New students can easily register online, reducing paperwork and administrative workload. The system guides students through the registration process and ensures that all required information is provided.

4. Efficient Attendance Tracking:

The SMS has automated attendance tracking, making it easier for teachers to record student attendance. Real-time attendance data is available, and automated notifications can be sent to parents and guardians for absent students, enhancing communication.

5. Academic Performance Monitoring:

Teachers can now monitor and assess student academic performance more effectively. The system allows for the recording of grades, generation of report cards, and analysis of student progress over time.

6. Improved Communication:

The SMS has facilitated better communication between the educational institution, teachers, and parents. Announcements, notifications, and updates can be shared through the system, ensuring that all stakeholders are well-informed.

7. Data Security:

The SMS has implemented robust security measures to protect student data. User authentication, role-based access control, and data encryption ensure that sensitive information is safeguarded against unauthorized access and data breaches.

8. Time and Cost Savings:

The automation of administrative tasks has resulted in significant time and cost savings for the educational institution. Manual data entry and paperwork have been reduced, allowing staff to focus on more strategic tasks.

9. Scalability:

The SMS has proven to be scalable, accommodating the growing student population and data volume. As the institution expands, the system can be easily scaled to meet the increasing demands.

10. User Satisfaction:

Users, including administrators, teachers, and students, have expressed high levels of satisfaction with the SMS. The user-friendly interface, accessibility, and efficiency have contributed to positive feedback.

11. Data Analysis:

The SMS provides tools for data analysis, allowing the institution to gain insights into student performance and trends. This data-driven approach enables evidence-based decision-making and improvements in academic strategies.

12. Documentation and Compliance:

The SMS has facilitated better documentation and compliance with data protection regulations. Detailed records of student information and system usage are maintained, ensuring accountability and adherence to legal requirements.

**Challenges Faced:**

The implementation of the Student Management System (SMS) was a transformative journey, but not without its set of challenges. Overcoming these obstacles was crucial to the successful deployment of the system. In this section, we outline some of the key challenges faced during the development and implementation of the SMS.

1. Resistance to Change:

One of the foremost challenges encountered was resistance to change among the stakeholders. Faculty and administrative staff were accustomed to traditional manual processes, and the introduction of an automated system required significant adjustment. Training and change management strategies were essential to overcome this challenge.

2. Data Migration:

Transferring existing student data from legacy systems to the new SMS posed a considerable challenge. Ensuring that data integrity was maintained during the migration process required meticulous planning and execution. Any data loss or corruption could have had severe consequences.

3. Technical Infrastructure:

The institution's existing technical infrastructure needed to be upgraded to support the SMS effectively. This included enhancements to network capabilities, server capacity, and compatibility with various devices and platforms. Ensuring a seamless technical transition was a complex task.

4. Data Security Concerns:

Safeguarding student data and maintaining data privacy were paramount. The SMS needed to comply with stringent data protection regulations, which presented a challenge in terms of implementing robust security measures, access controls, and encryption protocols.

5. User Training:

Training a diverse user base, including teachers, administrative staff, and students, was a substantial undertaking. Effective training programs had to be designed to ensure that all users could confidently navigate and utilize the SMS. Overcoming the digital literacy gap was also a challenge.

6. Customization and Integration:

Tailoring the SMS to meet the institution's specific needs required extensive customization. Additionally, integrating the SMS with other existing systems, such as Learning Management Systems (LMS) and email platforms, was complex. Compatibility issues and seamless data flow had to be addressed.

7. Scalability Planning:

Anticipating the growth of the institution and the increasing volume of student data was challenging. The SMS had to be designed to be scalable, allowing for expansion without significant disruptions. This necessitated careful architectural planning.

8. Resistance to Documentation:

Creating comprehensive documentation was met with resistance by the staff who were initially averse to meticulous record-keeping. Overcoming this challenge was vital for compliance with data protection regulations and for ensuring accountability.

9. Maintenance and Support:

Establishing an efficient maintenance and support system for the SMS was a challenge. This included addressing software updates, bug fixes, and user support. A dedicated team had to be in place to handle ongoing system maintenance.

10. Financial Constraints:

The budget allocated for the SMS project was limited, which presented a financial challenge. Balancing the desire for advanced features and customizations with budget constraints required careful financial management.

11. User Acceptance Testing:

Extensive User Acceptance Testing (UAT) was conducted to identify and resolve system issues, but coordinating this process was a challenge. Users had to allocate time for testing, and their feedback needed to be effectively integrated into the system.

12. Stakeholder Alignment:

Ensuring alignment among various stakeholders, including faculty, administration, and students, was an ongoing challenge.

**Conclusion:**

In the way our educational institution manages its student information and administrative tasks. This comprehensive system has brought about substantial improvements in data management, streamlining administrative processes, and enhancing overall efficiency. With the SMS in place, we have witnessed a remarkable transformation in how we handle student records and administrative tasks, leading to increased data accuracy and a reduction in manual errors.

One of the most notable advantages of the SMS is the way it has streamlined our administrative procedures. Our administrative staff can now perform tasks such as student admissions, record-keeping, and fee management with greater efficiency. The automation of these processes has led to a significant reduction in manual paperwork and administrative overhead, freeing up valuable time and resources.

Accessibility has also seen a remarkable improvement with the SMS. Being a web-based platform, it allows users to access the system from any device with an internet connection. This enhanced accessibility has made it convenient for both staff and students to interact with the system, improving communication and information sharing.

The security and privacy of student data have been of paramount importance, and the SMS has effectively addressed these concerns. Robust security measures have been implemented to protect student data, ensuring compliance with data protection regulations. This commitment to data security has instilled confidence in students, parents, and staff about the system's integrity.

The user-friendly interface of the SMS has made a significant impact, minimizing the learning curve for users. It has contributed to quick adoption and a smoother transition for all stakeholders. Additionally, the SMS offers powerful reporting and analytical tools that empower educators and administrators to make data-driven decisions, track student performance, and identify areas for improvement.

Integration capabilities have been seamlessly achieved, with the SMS integrated into other systems used within the institution, such as Learning Management Systems (LMS) and email platforms.

**Future Scope:**

The Student Management System (SMS) project, an innovative digital solution for educational institutions, has successfully streamlined various administrative tasks and student data management processes. Its implementation aimed to address the existing challenges associated with manual record-keeping, inefficient communication, and limited accessibility to student information. This report presents an overview of the project's journey, discussing the motivation behind its development, the objectives it aimed to achieve, the methodology employed, the challenges faced, results obtained, and its future scope.

The SMS project was initiated in response to the growing complexities in educational administration and the increasing need for an efficient, centralized system that could handle tasks such as student enrollment, data storage, performance tracking, and communication. The motivation behind the project stemmed from a commitment to enhancing the overall quality of education by reducing administrative burdens on faculty and staff.

The primary objective of the SMS was to create a user-friendly and comprehensive platform that would enable educational institutions to manage student information with ease, accuracy, and efficiency. By centralizing data and automating processes, the SMS aimed to eliminate redundancy, reduce errors, and provide quick access to critical data for informed decision-making.

The methodology employed for the development of the SMS included an in-depth analysis of the specific needs of the educational institution, followed by the design, implementation, and testing phases. The project's development team collaborated closely with faculty and administrative staff to ensure that the system was tailored to their requirements. During this phase, the integration of security measures and data encryption was a top priority to safeguard sensitive student information.

Challenges encountered during the project's implementation included data migration, user training, and resistance to change. Adapting to a digital system required the institution to migrate existing data accurately, ensuring that historical records were preserved. User training programs were essential to ensure that faculty and staff could efficiently navigate the new system. Overcoming resistance to change was achieved through effective communication and demonstrating the benefits of the SMS.

The results of implementing the SMS were transformative. Administrative processes that once required significant time and effort could now be completed efficiently. Student data became easily accessible and highly organized, leading to better decision-making and improved communication between faculty and students. The system's user-friendly interface and reporting capabilities further enhanced productivity.

Looking ahead, the future scope of the SMS is promising. The integration of a dedicated mobile application for students, parents, and faculty will enhance accessibility. Machine learning and analytics can be implemented to provide insights into student performance. Online examinations and assessments can be supported, offering flexibility and convenience. Additionally, alumni management, integration with Learning Management Systems (LMS), and financial management can further expand the SMS's capabilities.

In conclusion, the Student Management System has successfully addressed the challenges of educational administration and data management. By focusing on the project's motivation, objectives, methodology, and outcomes, this report provides an understanding of the system's significance. The SMS not only streamlines administrative tasks but also contributes to the overall improvement of the educational experience for students and educators. Its future scope promises further innovation and digital transformation within the educational institution.

**Frontend and Backend codes:**

from tkinter import \*

import time

import ttkthemes

from tkinter import ttk,messagebox,filedialog

import pymysql

import pandas

#functionality Part

def iexit():

    result=messagebox.askyesno('Confirm','Do you want to exit?')

    if result:

        root.destroy()

    else:

        pass

def export\_data():

    url=filedialog.asksaveasfilename(defaultextension='.csv')

    indexing=studentTable.get\_children()

    newlist=[]

    for index in indexing:

        content=studentTable.item(index)

        datalist=content['values']

        newlist.append(datalist)

    table=pandas.DataFrame(newlist,columns=['Id','Name','Mobile','Email','Address','Gender','DOB','Added Date','Added Time'])

    table.to\_csv(url,index=False)

    messagebox.showinfo('Success','Data is saved succesfully')

def toplevel\_data(title,button\_text,command):

    global idEntry,phoneEntry,nameEntry,emailEntry,addressEntry,genderEntry,dobEntry,screen

    screen = Toplevel()

    screen.title(title)

    screen.grab\_set()

    screen.resizable(False, False)

    idLabel = Label(screen, text='Id', font=('times new roman', 20, 'bold'))

    idLabel.grid(row=0, column=0, padx=30, pady=15, sticky=W)

    idEntry = Entry(screen, font=('roman', 15, 'bold'), width=24)

    idEntry.grid(row=0, column=1, pady=15, padx=10)

    nameLabel = Label(screen, text='Name', font=('times new roman', 20, 'bold'))

    nameLabel.grid(row=1, column=0, padx=30, pady=15, sticky=W)

    nameEntry = Entry(screen, font=('roman', 15, 'bold'), width=24)

    nameEntry.grid(row=1, column=1, pady=15, padx=10)

    phoneLabel = Label(screen, text='Phone', font=('times new roman', 20, 'bold'))

    phoneLabel.grid(row=2, column=0, padx=30, pady=15, sticky=W)

    phoneEntry = Entry(screen, font=('roman', 15, 'bold'), width=24)

    phoneEntry.grid(row=2, column=1, pady=15, padx=10)

    emailLabel = Label(screen, text='Email', font=('times new roman', 20, 'bold'))

    emailLabel.grid(row=3, column=0, padx=30, pady=15, sticky=W)

    emailEntry = Entry(screen, font=('roman', 15, 'bold'), width=24)

    emailEntry.grid(row=3, column=1, pady=15, padx=10)

    addressLabel = Label(screen, text='Address', font=('times new roman', 20, 'bold'))

    addressLabel.grid(row=4, column=0, padx=30, pady=15, sticky=W)

    addressEntry = Entry(screen, font=('roman', 15, 'bold'), width=24)

    addressEntry.grid(row=4, column=1, pady=15, padx=10)

    genderLabel = Label(screen, text='Gender', font=('times new roman', 20, 'bold'))

    genderLabel.grid(row=5, column=0, padx=30, pady=15, sticky=W)

    genderEntry = Entry(screen, font=('roman', 15, 'bold'), width=24)

    genderEntry.grid(row=5, column=1, pady=15, padx=10)

    dobLabel = Label(screen, text='D.O.B', font=('times new roman', 20, 'bold'))

    dobLabel.grid(row=6, column=0, padx=30, pady=15, sticky=W)

    dobEntry = Entry(screen, font=('roman', 15, 'bold'), width=24)

    dobEntry.grid(row=6, column=1, pady=15, padx=10)

    student\_button = ttk.Button(screen, text=button\_text, command=command)

    student\_button.grid(row=7, columnspan=2, pady=15)

    if title=='Update Student':

        indexing = studentTable.focus()

        content = studentTable.item(indexing)

        listdata = content['values']

        idEntry.insert(0, listdata[0])

        nameEntry.insert(0, listdata[1])

        phoneEntry.insert(0, listdata[2])

        emailEntry.insert(0, listdata[3])

        addressEntry.insert(0, listdata[4])

        genderEntry.insert(0, listdata[5])

        dobEntry.insert(0, listdata[6])

def update\_data():

    query='update student set name=%s,mobile=%s,email=%s,address=%s,gender=%s,dob=%s,date=%s,time=%s where id=%s'

    mycursor.execute(query,(nameEntry.get(),phoneEntry.get(),emailEntry.get(),addressEntry.get(),

                            genderEntry.get(),dobEntry.get(),date,currenttime,idEntry.get()))

    con.commit()

    messagebox.showinfo('Success',f'Id {idEntry.get()} is modified successfully',parent=screen)

    screen.destroy()

    show\_student()

def show\_student():

    query = 'select \* from student'

    mycursor.execute(query)

    fetched\_data = mycursor.fetchall()

    studentTable.delete(\*studentTable.get\_children())

    for data in fetched\_data:

        studentTable.insert('', END, values=data)

def delete\_student():

    indexing=studentTable.focus()

    print(indexing)

    content=studentTable.item(indexing)

    content\_id=content['values'][0]

    query='delete from student where id=%s'

    mycursor.execute(query,content\_id)

    con.commit()

    messagebox.showinfo('Deleted',f'Id {content\_id} is deleted succesfully')

    query='select \* from student'

    mycursor.execute(query)

    fetched\_data=mycursor.fetchall()

    studentTable.delete(\*studentTable.get\_children())

    for data in fetched\_data:

        studentTable.insert('',END,values=data)

def search\_data():

    query='select \* from student where id=%s or name=%s or email=%s or mobile=%s or address=%s or gender=%s or dob=%s'

    mycursor.execute(query,(idEntry.get(),nameEntry.get(),emailEntry.get(),phoneEntry.get(),addressEntry.get(),genderEntry.get(),dobEntry.get()))

    studentTable.delete(\*studentTable.get\_children())

    fetched\_data=mycursor.fetchall()

    for data in fetched\_data:

        studentTable.insert('',END,values=data)

def add\_data():

    if idEntry.get()=='' or nameEntry.get()=='' or phoneEntry.get()=='' or emailEntry.get()=='' or addressEntry.get()=='' or genderEntry.get()=='' or dobEntry.get()=='':

        messagebox.showerror('Error','All Feilds are required',parent=screen)

    else:

        try:

            query='insert into student values(%s,%s,%s,%s,%s,%s,%s,%s,%s)'

            mycursor.execute(query,(idEntry.get(),nameEntry.get(),phoneEntry.get(),emailEntry.get(),addressEntry.get(),

                                    genderEntry.get(),dobEntry.get(),date,currenttime))

            con.commit()

            result=messagebox.askyesno('Confirm','Data added successfully. Do you want to clean the form?',parent=screen)

            if result:

                idEntry.delete(0,END)

                nameEntry.delete(0,END)

                phoneEntry.delete(0,END)

                emailEntry.delete(0,END)

                addressEntry.delete(0,END)

                genderEntry.delete(0,END)

                dobEntry.delete(0,END)

            else:

                pass

        except:

            messagebox.showerror('Error','Id cannot be repeated',parent=screen)

            return

        query='select \*from student'

        mycursor.execute(query)

        fetched\_data=mycursor.fetchall()

        studentTable.delete(\*studentTable.get\_children())

        for data in fetched\_data:

            studentTable.insert('',END,values=data)

def connect\_database():

    def connect():

        global mycursor,con

        try:

            con=pymysql.connect(host='localhost',user='root',password='damon')

            mycursor=con.cursor()

        except:

            messagebox.showerror('Error','Invalid Details',parent=connectWindow)

            return

        try:

            query='create database studentmanagementsystem'

            mycursor.execute(query)

            query='use studentmanagementsystem'

            mycursor.execute(query)

            query='create table student(id int not null primary key, name varchar(30),mobile varchar(10),email varchar(30),' \

                  'address varchar(100),gender varchar(20),dob varchar(20),date varchar(50), time varchar(50))'

            mycursor.execute(query)

        except:

            query='use studentmanagementsystem'

            mycursor.execute(query)

        messagebox.showinfo('Success', 'Database Connection is successful', parent=connectWindow)

        connectWindow.destroy()

        addstudentButton.config(state=NORMAL)

        searchstudentButton.config(state=NORMAL)

        updatestudentButton.config(state=NORMAL)

        showstudentButton.config(state=NORMAL)

        exportstudentButton.config(state=NORMAL)

        deletestudentButton.config(state=NORMAL)

    connectWindow=Toplevel()

    connectWindow.grab\_set()

    connectWindow.geometry('470x250+730+230')

    connectWindow.title('Database Connection')

    connectWindow.resizable(0,0)

    hostnameLabel=Label(connectWindow,text='Host Name',font=('arial',20,'bold'))

    hostnameLabel.grid(row=0,column=0,padx=20)

    hostEntry=Entry(connectWindow,font=('roman',15,'bold'),bd=2)

    hostEntry.grid(row=0,column=1,padx=40,pady=20)

    usernameLabel = Label(connectWindow, text='User Name', font=('arial', 20, 'bold'))

    usernameLabel.grid(row=1, column=0, padx=20)

    usernameEntry = Entry(connectWindow, font=('roman', 15, 'bold'), bd=2)

    usernameEntry.grid(row=1, column=1, padx=40, pady=20)

    passwordLabel = Label(connectWindow, text='Password', font=('arial', 20, 'bold'))

    passwordLabel.grid(row=2, column=0, padx=20)

    passwordEntry = Entry(connectWindow, font=('roman', 15, 'bold'), bd=2)

    passwordEntry.grid(row=2, column=1, padx=40, pady=20)

    connectButton=ttk.Button(connectWindow,text='CONNECT',command=connect)

    connectButton.grid(row=3,columnspan=2)

count=0

text=''

def slider():

    global text,count

    # if count==len(s):

    #     count=0

    #     text=''

    text=text+s[count]

    sliderLabel.config(text=text)

    count+=1

    sliderLabel.after(300,slider)

def clock():

    global date,currenttime

    date=time.strftime('%d/%m/%Y')

    currenttime=time.strftime('%H:%M:%S')

    datetimeLabel.config(text=f'   Date: {date}\nTime: {currenttime}')

    datetimeLabel.after(1000,clock)

#GUI Part

root=ttkthemes.ThemedTk()

root.get\_themes()

root.set\_theme('radiance')

root.geometry('1174x680+0+0')

root.resizable(0,0)

root.title('Student Management System')

datetimeLabel=Label(root,font=('times new roman',18,'bold'))

datetimeLabel.place(x=5,y=5)

clock()

s='Student Management System' #s[count]=t when count is 1

sliderLabel=Label(root,font=('arial',28,'italic bold'),width=30)

sliderLabel.place(x=200,y=0)

slider()

connectButton=ttk.Button(root,text='Connect database',command=connect\_database)

connectButton.place(x=980,y=0)

leftFrame=Frame(root)

leftFrame.place(x=50,y=80,width=300,height=600)

logo\_image=PhotoImage(file='student.png')

logo\_Label=Label(leftFrame,image=logo\_image)

logo\_Label.grid(row=0,column=0)

addstudentButton=ttk.Button(leftFrame,text='Add Student',width=25,state=DISABLED,command=lambda :toplevel\_data('Add Student','Add',add\_data))

addstudentButton.grid(row=1,column=0,pady=20)

searchstudentButton=ttk.Button(leftFrame,text='Search Student',width=25,state=DISABLED,command=lambda :toplevel\_data('Search Student','Search',search\_data))

searchstudentButton.grid(row=2,column=0,pady=20)

deletestudentButton=ttk.Button(leftFrame,text='Delete Student',width=25,state=DISABLED,command=delete\_student)

deletestudentButton.grid(row=3,column=0,pady=20)

updatestudentButton=ttk.Button(leftFrame,text='Update Student',width=25,state=DISABLED,command=lambda :toplevel\_data('Update Student','Update',update\_data))

updatestudentButton.grid(row=4,column=0,pady=20)

showstudentButton=ttk.Button(leftFrame,text='Show Student',width=25,state=DISABLED,command=show\_student)

showstudentButton.grid(row=5,column=0,pady=20)

exportstudentButton=ttk.Button(leftFrame,text='Export data',width=25,state=DISABLED,command=export\_data)

exportstudentButton.grid(row=6,column=0,pady=20)

exitButton=ttk.Button(leftFrame,text='Exit',width=25,command=iexit)

exitButton.grid(row=7,column=0,pady=20)

rightFrame=Frame(root)

rightFrame.place(x=350,y=80,width=820,height=600)

scrollBarX=Scrollbar(rightFrame,orient=HORIZONTAL)

scrollBarY=Scrollbar(rightFrame,orient=VERTICAL)

studentTable=ttk.Treeview(rightFrame,columns=('Id','Name','Mobile','Email','Address','Gender',

                                 'D.O.B','Added Date','Added Time'),

                          xscrollcommand=scrollBarX.set,yscrollcommand=scrollBarY.set)

scrollBarX.config(command=studentTable.xview)

scrollBarY.config(command=studentTable.yview)

scrollBarX.pack(side=BOTTOM,fill=X)

scrollBarY.pack(side=RIGHT,fill=Y)

studentTable.pack(expand=1,fill=BOTH)

studentTable.heading('Id',text='Id')

studentTable.heading('Name',text='Name')

studentTable.heading('Mobile',text='Mobile No')

studentTable.heading('Email',text='Email Address')

studentTable.heading('Address',text='Address')

studentTable.heading('Gender',text='Gender')

studentTable.heading('D.O.B',text='D.O.B')

studentTable.heading('Added Date',text='Added Date')

studentTable.heading('Added Time',text='Added Time')

studentTable.column('Id',width=50,anchor=CENTER)

studentTable.column('Name',width=200,anchor=CENTER)

studentTable.column('Email',width=300,anchor=CENTER)

studentTable.column('Mobile',width=200,anchor=CENTER)

studentTable.column('Address',width=300,anchor=CENTER)

studentTable.column('Gender',width=100,anchor=CENTER)

studentTable.column('D.O.B',width=200,anchor=CENTER)

studentTable.column('Added Date',width=200,anchor=CENTER)

studentTable.column('Added Time',width=200,anchor=CENTER)

style=ttk.Style()

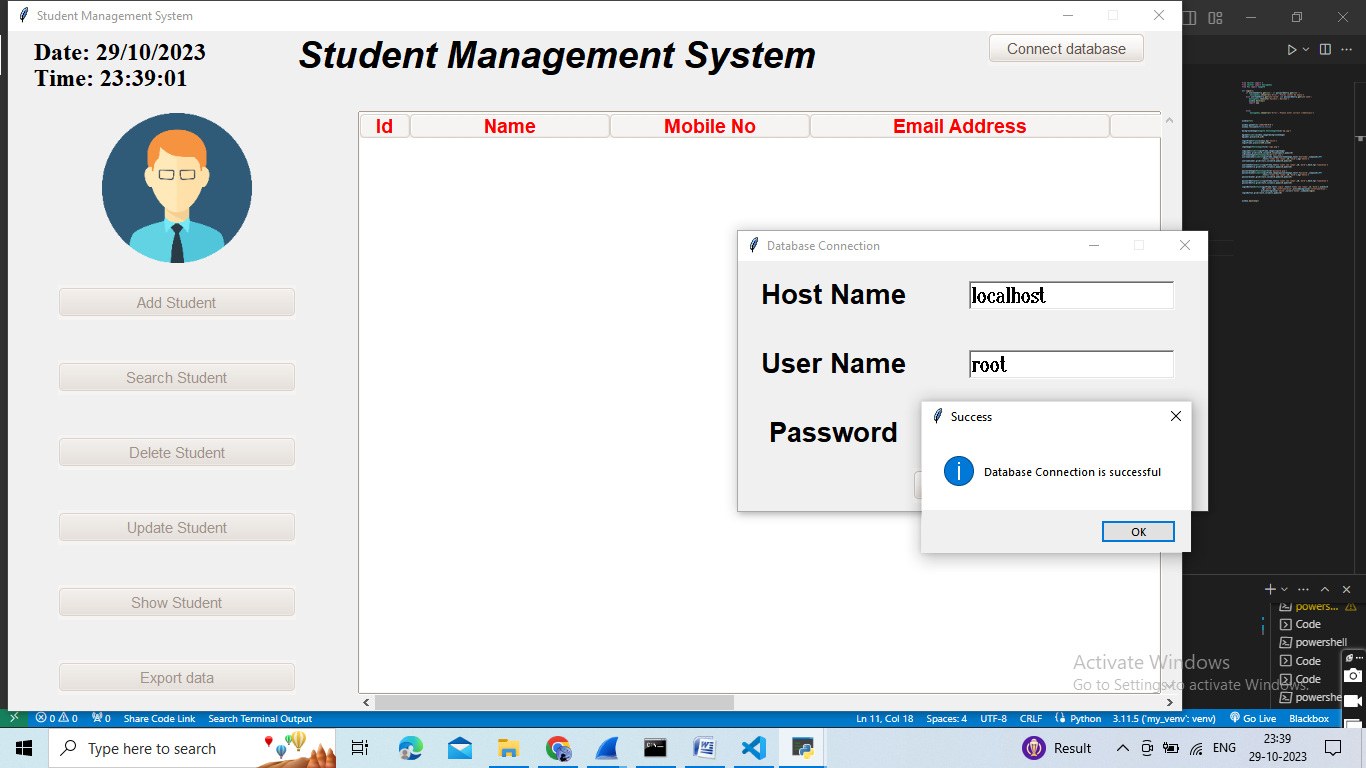
style.configure('Treeview', rowheight=40,font=('arial', 12, 'bold'), fieldbackground='white', background='white',)

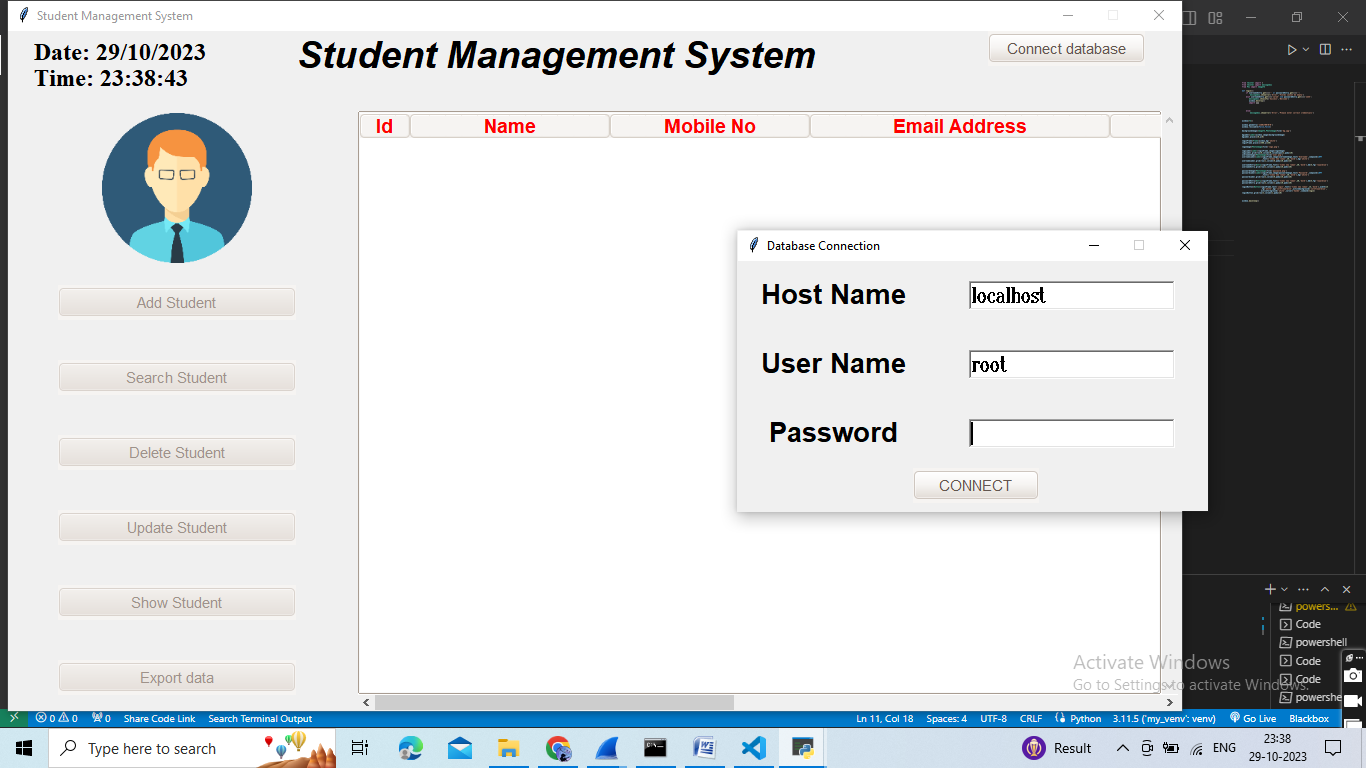
style.configure('Treeview.Heading',font=('arial', 14, 'bold'),foreground='red')

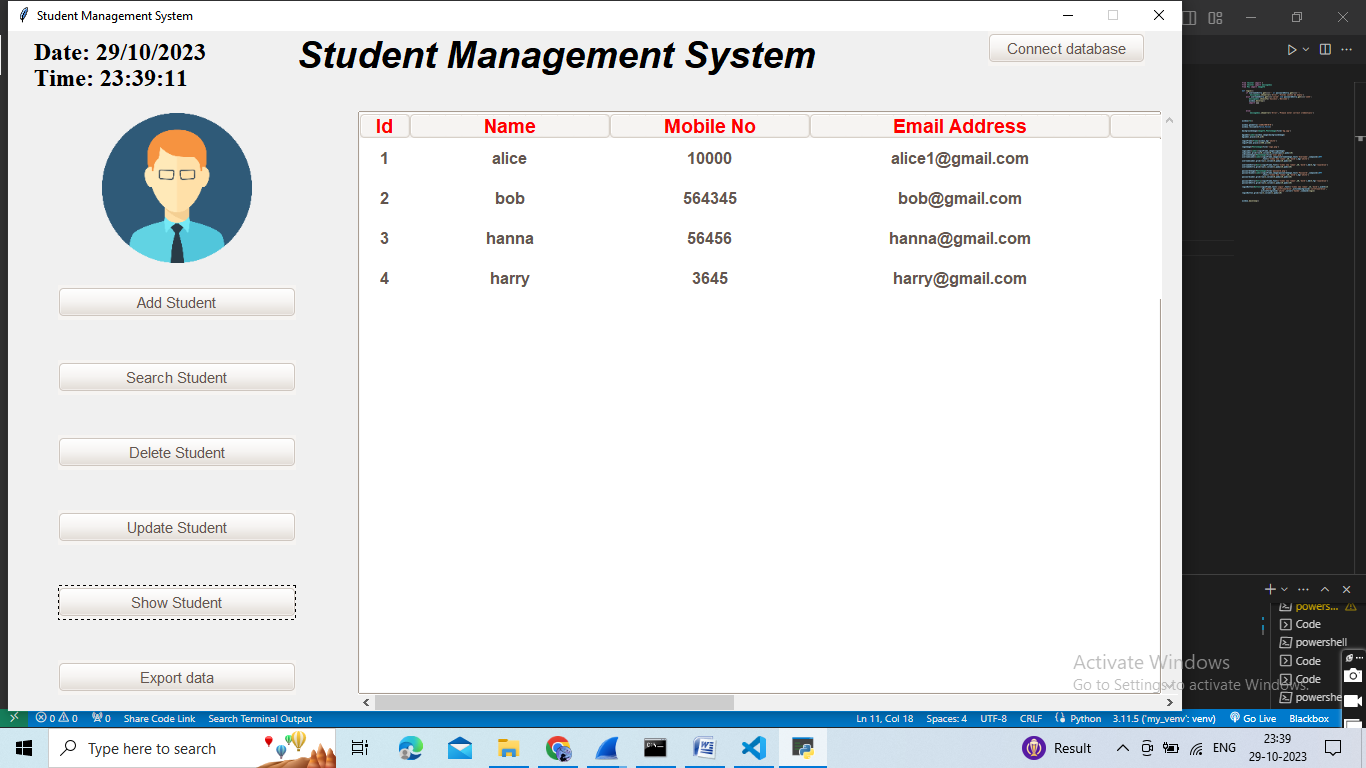
studentTable.config(show='headings')

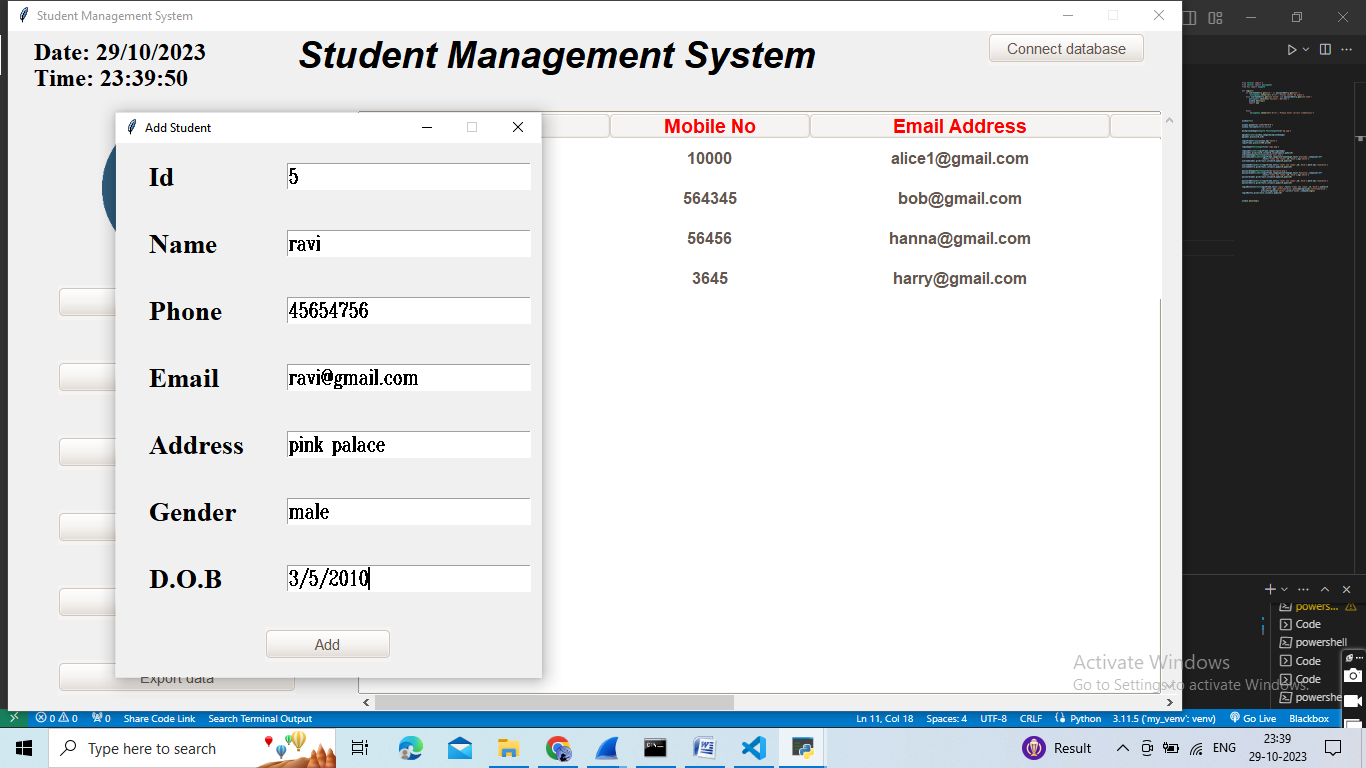
root.mainloop()

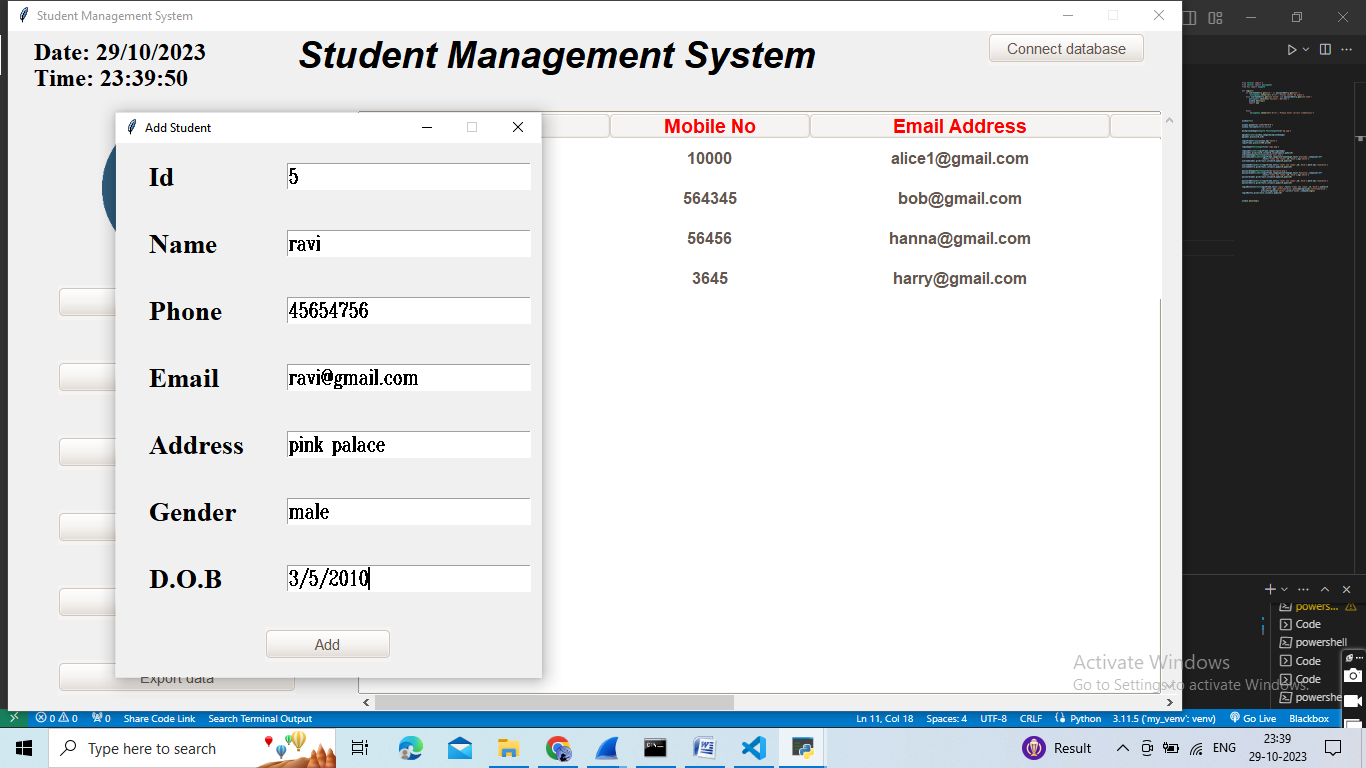
**Output:**











**References**

* Student Management System in Python by GeeksforGeeks: https://www.geeksforgeeks.org/student-management-system-in-python/
* A Student Management System with MySQL | Login System by Dev Ed: https://www.exploit-db.com/exploits/48896
* Student Management System Project In Python by CopyAssignment: https://copyassignment.com/student-management-system-project-in-python/
* Student Management System with MySQL by CodeWithMosh: https://www.youtube.com/watch?v=ohb-oknZQa8
* Student Management System with Python and MySQL by Tutorials Point: https://m.youtube.com/watch?v=X4iITVCyQlw