

Student Management System with Python and MongoDB Integration

Ashish Sharma
Data Analyst

Abstract

The Student Management System (SMS) is a Python application designed to streamline the process of managing student records in educational institutions. Leveraging the Tkinter library for the graphical user interface (GUI) and MongoDB for data storage, the SMS offers a user-friendly platform for administrators to add, view, update, and delete student information. By connecting to a MongoDB database, the application ensures data integrity and scalability, while the intuitive GUI enhances usability. Through this project report, the architecture, functionalities, and usage examples of the SMS are detailed, highlighting its significance in modernizing student data management processes.

Introduction

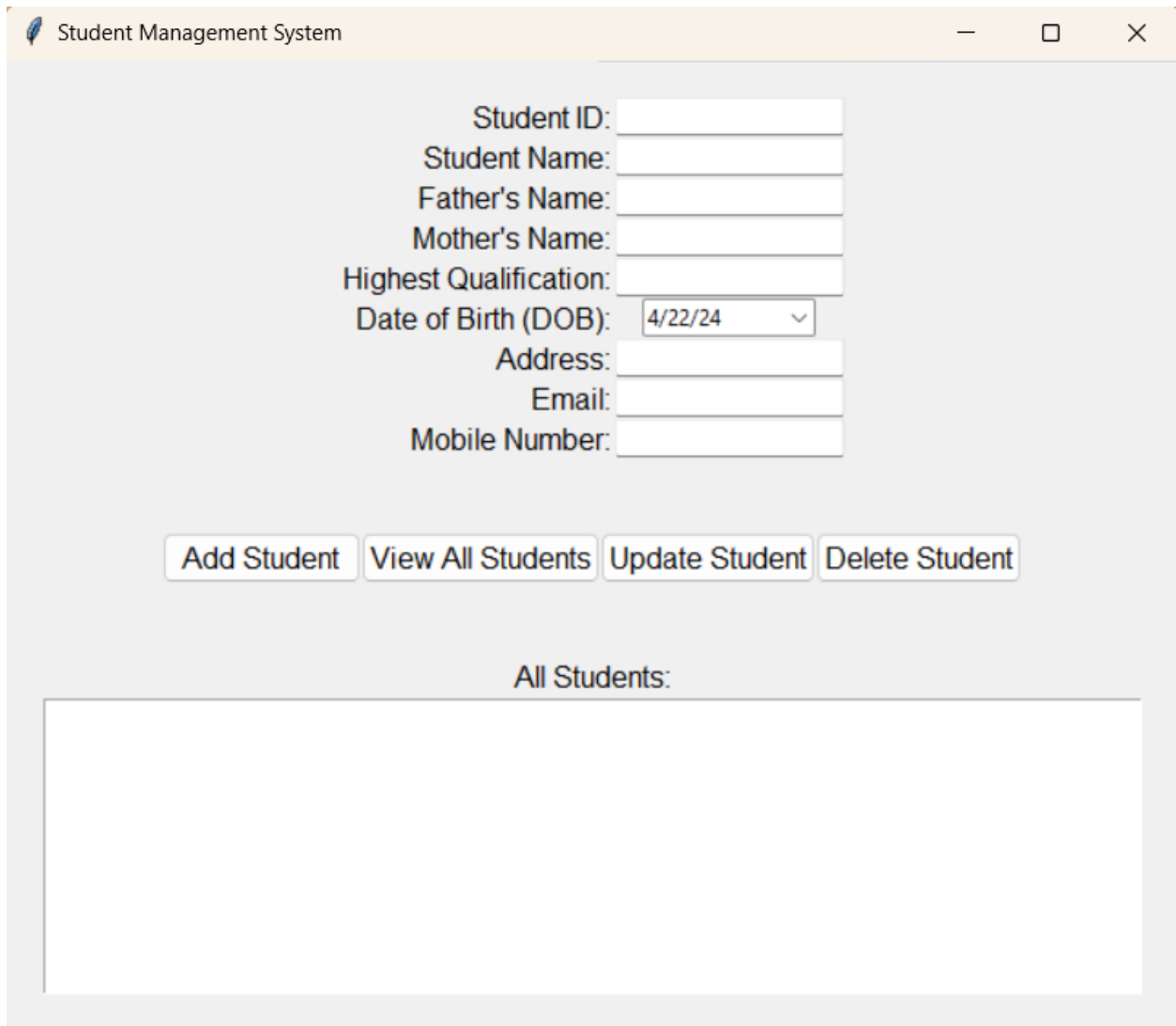
In educational institutions, managing student data efficiently is paramount for ensuring smooth administrative operations. Traditionally, this process relied on manual record-keeping systems, which were not only time-consuming but also prone to errors and inconsistencies. Recognizing the need for a more streamlined approach, the Student Management System (SMS) was developed as a digital solution to address these challenges.

The SMS is a Python application that provides a centralized platform for storing, accessing, and managing student records. Built using the Tkinter library, the application offers an intuitive graphical user interface (GUI) that simplifies the input and retrieval of student information. Moreover, by utilizing MongoDB as the backend database, the SMS ensures data integrity, scalability, and flexibility, making it suitable for institutions of varying sizes and complexities.

Through this project report, we delve into the architecture, functionalities, and usage examples of the SMS, demonstrating its significance in modernizing student data management processes. By leveraging the power of Python, Tkinter, and MongoDB, the SMS empowers educational institutions to streamline their administrative workflows, enhance data accuracy, and ultimately, focus more on delivering quality education.

Working

The application connects to a MongoDB database hosted on MongoDB Atlas, a cloud-based database service. It establishes a connection using the PyMongo library and interacts with the "student_management" database, which contains a collection named "students" to store student records.



The screenshot displays the 'Student Management System' window. It features a form with the following fields: 'Student ID:', 'Student Name:', 'Father's Name:', 'Mother's Name:', 'Highest Qualification:', 'Date of Birth (DOB):' (with a dropdown menu showing '4/22/24'), 'Address:', 'Email:', and 'Mobile Number:'. Below the form are four buttons: 'Add Student', 'View All Students', 'Update Student', and 'Delete Student'. At the bottom, there is a section labeled 'All Students:' followed by a large, empty rectangular box for displaying student records.

Upon launching the application, users are presented with a graphical user interface (GUI) created using Tkinter. The GUI includes entry fields for entering student information such as student ID, name, father's name, mother's name, highest qualification, date of birth (DOB), address, email, and mobile number. Users can add new student records by filling in these fields and clicking the "Add Student" button. The application validates the input and adds the student record to the MongoDB database upon successful validation.

The application also provides functionality to view all student records stored in the database. Users can click the "View All Students" button to retrieve and display all student records in a list format on the GUI.

Additionally, users can update existing student records by selecting a record from the list, modifying the desired fields, and clicking the "Update Student" button. Similarly, users can delete student records by selecting a record from the list and clicking the "Delete Student" button.

The GUI layout is designed for ease of use, with labeled entry fields and buttons organized in a logical manner. Error messages are displayed using message boxes to prompt users to correct any input errors or notify them of successful operations.

Code Explanation

Code:-

```
import tkinter as tk
from tkinter import ttk, messagebox
from tkcalendar import DateEntry # Import DateEntry widget from tkcalendar
import pymongo

# Connect to MongoDB
client =
pymongo.MongoClient("mongodb+srv://ashish:ashish@cluster0.ztjk4er.mongodb.net/")
db = client["student_management"]
students_col = db["students"]

# Declare global variables for entry fields
id_entry = None
name_entry = None
father_name_entry = None
mother_name_entry = None
highest_qualification_entry = None
dob_entry = None
address_entry = None
email_entry = None
mobile_entry = None

def add_student():
    global id_entry, name_entry, father_name_entry, mother_name_entry,
highest_qualification_entry, dob_entry, address_entry, email_entry, mobile_entry
    student_id = id_entry.get()
    name = name_entry.get()
    father_name = father_name_entry.get()
```

```

mother_name = mother_name_entry.get()
highest_qualification = highest_qualification_entry.get()
dob = dob_entry.get() # For DateEntry widget, use get_date() instead of
get()
address = address_entry.get()
email = email_entry.get()
mobile = mobile_entry.get()

if student_id and name and father_name and mother_name and
highest_qualification and dob and address and email and mobile:
    student_data = {
        "_id": student_id,
        "name": name,
        "father_name": father_name,
        "mother_name": mother_name,
        "highest_qualification": highest_qualification,
        "dob": dob,
        "address": address,
        "email": email,
        "mobile": mobile
    }
    students_col.insert_one(student_data)
    messagebox.showinfo("Success", "Student added successfully.")
    clear_entries()
else:
    messagebox.showerror("Error", "Please fill in all fields.")

def clear_entries():
    global id_entry, name_entry, father_name_entry, mother_name_entry,
highest_qualification_entry, dob_entry, address_entry, email_entry, mobile_entry
    id_entry.delete(0, tk.END)
    name_entry.delete(0, tk.END)
    father_name_entry.delete(0, tk.END)
    mother_name_entry.delete(0, tk.END)
    highest_qualification_entry.delete(0, tk.END)
    dob_entry.set_date(None) # Clear DateEntry widget
    address_entry.delete(0, tk.END)
    email_entry.delete(0, tk.END)
    mobile_entry.delete(0, tk.END)

def view_students():
    students = students_col.find()
    if students:
        student_list.delete(0, tk.END) # Clear previous entries
        for student in students:
            student_info = f"ID: {student.get('_id', '')}, Name:
{student.get('name', '')}, Father's Name: {student.get('father_name', '')},
Mother's Name: {student.get('mother_name', '')}, Highest Qualification:

```

```

{student.get('highest_qualification', ''), DOB: {student.get('dob', '')},
Address: {student.get('address', '')}, Email: {student.get('email', '')}, Mobile:
{student.get('mobile', '')}}"
        student_list.insert(tk.END, student_info)
    else:
        messagebox.showinfo("No Students", "No students found.")

def update_student():
    global id_entry, name_entry
    student_id = id_entry.get()
    new_name = name_entry.get()
    if student_id and new_name:
        students_col.update_one({"_id": student_id}, {"$set": {"name":
new_name}})
        messagebox.showinfo("Success", "Student updated successfully.")
        clear_entries()
    else:
        messagebox.showerror("Error", "Please fill in all fields.")

def delete_student():
    student_id = id_entry.get()
    if student_id:
        result = students_col.delete_one({"_id": student_id})
        if result.deleted_count > 0:
            messagebox.showinfo("Success", "Student deleted successfully.")
            clear_entries()
        else:
            messagebox.showerror("Error", "Student not found.")
    else:
        messagebox.showerror("Error", "Please fill in all fields.")

def main():
    global id_entry, name_entry, father_name_entry, mother_name_entry,
highest_qualification_entry, dob_entry, address_entry, email_entry, mobile_entry,
student_list
    root = tk.Tk()
    root.title("Student Management System")

    style = ttk.Style()
    style.configure("TLabel", font=("Arial", 12))
    style.configure("TEntry", font=("Arial", 12))
    style.configure("TButton", font=("Arial", 12))

    label_frame = ttk.Frame(root, padding="20")
    label_frame.grid(row=0, column=0, columnspan=2)

    id_label = ttk.Label(label_frame, text="Student ID:")
    id_label.grid(row=0, column=0, sticky="e")

```

```

id_entry = ttk.Entry(label_frame)
id_entry.grid(row=0, column=1)

name_label = ttk.Label(label_frame, text="Student Name:")
name_label.grid(row=1, column=0, sticky="e")
name_entry = ttk.Entry(label_frame)
name_entry.grid(row=1, column=1)

father_name_label = ttk.Label(label_frame, text="Father's Name:")
father_name_label.grid(row=2, column=0, sticky="e")
father_name_entry = ttk.Entry(label_frame)
father_name_entry.grid(row=2, column=1)

mother_name_label = ttk.Label(label_frame, text="Mother's Name:")
mother_name_label.grid(row=3, column=0, sticky="e")
mother_name_entry = ttk.Entry(label_frame)
mother_name_entry.grid(row=3, column=1)

highest_qualification_label = ttk.Label(label_frame, text="Highest
Qualification:")
highest_qualification_label.grid(row=4, column=0, sticky="e")
highest_qualification_entry = ttk.Entry(label_frame)
highest_qualification_entry.grid(row=4, column=1)

dob_label = ttk.Label(label_frame, text="Date of Birth (DOB):")
dob_label.grid(row=5, column=0, sticky="e")
dob_entry = DateEntry(label_frame, width=12, background='darkblue',
foreground='white', borderwidth=2)
dob_entry.grid(row=5, column=1)

address_label = ttk.Label(label_frame, text="Address:")
address_label.grid(row=6, column=0, sticky="e")
address_entry = ttk.Entry(label_frame)
address_entry.grid(row=6, column=1)

email_label = ttk.Label(label_frame, text="Email:")
email_label.grid(row=7, column=0, sticky="e")
email_entry = ttk.Entry(label_frame)
email_entry.grid(row=7, column=1)

mobile_label = ttk.Label(label_frame, text="Mobile Number:")
mobile_label.grid(row=8, column=0, sticky="e")
mobile_entry = ttk.Entry(label_frame)
mobile_entry.grid(row=8, column=1)

button_frame = ttk.Frame(root, padding="20")
button_frame.grid(row=1, column=0, columnspan=2)

```

```

    add_button = ttk.Button(button_frame, text="Add Student",
command=add_student)
    add_button.grid(row=0, column=0)

    view_button = ttk.Button(button_frame, text="View All Students",
command=view_students)
    view_button.grid(row=0, column=1)

    update_button = ttk.Button(button_frame, text="Update Student",
command=update_student)
    update_button.grid(row=0, column=2)

    delete_button = ttk.Button(button_frame, text="Delete Student",
command=delete_student)
    delete_button.grid(row=0, column=3)

    student_list_frame = ttk.Frame(root, padding="20")
    student_list_frame.grid(row=2, column=0, columnspan=2)

    student_list_label = ttk.Label(student_list_frame, text="All Students:")
    student_list_label.grid(row=0, column=0, columnspan=2)

    student_list = tk.Listbox(student_list_frame, width=100) # Increased width
for better visibility
    student_list.grid(row=1, column=0, columnspan=2)

    root.mainloop()

if __name__ == "__main__":
    main()

```

Explanation: -

This Python script implements a Student Management System (SMS) using the Tkinter library for GUI development and MongoDB for data storage. Below is a detailed explanation of each component and functionality of the code:

1. Import Statements:

- The script starts by importing necessary libraries:
 - **tkinter** and **ttk** for GUI development.
 - **messagebox** for displaying messages.
 - **DateEntry** from **tkcalendar** for date input.
 - **pymongo** for connecting to MongoDB.

2. Connect to MongoDB:

- The script establishes a connection to a MongoDB database hosted on MongoDB Atlas using **pymongo.MongoClient**. The database name is set to **"student_management"**, and a collection named **"students"** is accessed.

3. Global Variables:

- Global variables are declared for entry fields (**id_entry**, **name_entry**, etc.) to store references to Tkinter Entry widgets.

4. Function Definitions:

- Several functions are defined to perform CRUD operations on student records:
 - **add_student()**: Adds a new student record to the database.
 - **clear_entries()**: Clears all entry fields.
 - **view_students()**: Retrieves and displays all student records in a listbox.
 - **update_student()**: Updates an existing student record in the database.
 - **delete_student()**: Deletes a student record from the database.

5. Main Function (main()):

- The **main()** function creates the GUI layout using Tkinter widgets:
 - Labels and entry fields for inputting student information.
 - Buttons for adding, viewing, updating, and deleting student records.
 - A listbox to display all student records.
- The **mainloop()** method is called to start the Tkinter event loop.

6. Styling with ttk.Style():

- Styling options are configured for labels, entry fields, and buttons using **ttk.Style()**.

7. Event Handlers:

- Event handlers are attached to buttons (**add_button**, **view_button**, etc.) to trigger corresponding functions when clicked.

8. Error Handling:

- Error messages are displayed using **messagebox** to notify users of input errors or successful operations.

9. Run Application:

- The **main()** function is called if the script is executed directly (**__name__ == "__main__"**), initiating the SMS application.

Examples of Performing Operations

1. Adding a student:

- Enter student information in the provided entry fields.
- Click the "Add Student" button to add the student record to the database.
- Upon successful addition, a message box will confirm the operation, and the entry fields will be cleared for the next entry.

The screenshot shows a web form for adding a student. The form fields are: Student ID (23MSM40003), Student Name (Pamanji Jagadeesh), Father's Name (Pamanji), Mother's Name (empty), Highest Qualification (BTech), Date of Birth (DOB) (4/22/99), Address (Andhra Pradesh), Email (eesh1999@gmail.com), and Mobile Number (9059468456). Below the form are four buttons: Add Student, View All Students, Update Student, and Delete Student. To the right of the form is a success message box titled 'Success' with a blue information icon and the text 'Student added successfully.' and an OK button.

2. Viewing All Students:

- Click the "View All Students" button to retrieve and display all student records from the database.
-
- The listbox will show the details of all students, including their IDs, names, parents' names, qualifications, DOBs, addresses, emails, and mobile numbers.

All Students:									
ID: 12,	Name: asdf,	Father's Name: ,	Mother's Name: ,	Highest Qualification: ,	DOB: ,	Address: ,	Email: ,	Mobile: ,	
ID: asdsad,	Name: eqwfqw,	Father's Name: ,	Mother's Name: ,	Highest Qualification: ,	DOB: ,	Address: ,	Email: ,	Mo: ,	
ID: 23msm40001,	Name: Ashish,	Father's Name: ,	Mother's Name: ,	Highest Qualification: ,	DOB: ,	Address: ,	Email: ,	Mobile: ,	
ID: 23msm40002,	Name: Ashish,	Father's Name: Kamlesh,	Mother's Name: Nirmla,	Highest Qualification: BCA,	DOI: ,	Address: ,	Email: ,	Mobile: ,	
ID: 123123,	Name: dfsf,	Father's Name: sdfdsdfs,	Mother's Name: dfsfsdf,	Highest Qualification: sdfdf,	DOB: 4/9/2000,	Address: ,	Email: ,	Mobile: ,	
ID: 23msm40089,	Name: Komalpreet Kaur,	Father's Name: Gurbachan Singh,	Mother's Name: Pata Nahi,	Highest Qualification: ,	DOB: ,	Address: ,	Email: ,	Mobile: ,	
ID: 23MSM40003,	Name: Pamanji Jagadeesh,	Father's Name: Pamanji,	Mother's Name: Pamanji,	Highest Qualification: BTech,	DOB: 4/22/1999,	Address: Andhra Pradesh,	Email: eesh1999@gmail.com,	Mobile: 9059468456,	

3. Updating a student:

- Select a student record from the listbox by clicking on it.
- Modify the desired fields in the entry fields.
- Click the "Update Student" button to update the student record in the database.
- Upon successful update, a message box will confirm the operation, and the entry fields will be cleared.

Student ID:
 Student Name:
 Father's Name:
 Mother's Name:
 Highest Qualification:
 Date of Birth (DOB):
 Address:
 Email:
 Mobile Number:

All Students:

ID: 12, Name: asdf, Father's Name: , Mother's Name: , Highest Qualification: , DOB: , Address: , Email: , Mobile:
ID: asdsad, Name: eqwfw, Father's Name: , Mother's Name: , Highest Qualification: , DOB: , Address: , Email: , Mo
ID: 23msm40001, Name: Ashish, Father's Name: , Mother's Name: , Highest Qualification: , DOB: , Address: , Email:
ID: 23msm40002, Name: Ashish, Father's Name: Kamlesh, Mother's Name: Nirmla, Highest Qualification: BCA, DOI
ID: 123123, Name: dfsf, Father's Name: sdfdsdfs, Mother's Name: dfsfsdf, Highest Qualification: sdfdf, DOB: 4/9/2
ID: 23msm40089, Name: Komalpreet Kaur, Father's Name: Gurbachan Singh, Mother's Name: Pata Nahi, Highest Q
ID: 23MSM40003, Name: Pamanji Jagadeesh, Father's Name: Pamanji, Mother's Name: Pamanji, Highest Qualificat

Success

Student updated successfully.

OK

4. Deleting a student:

- Select a student record from the list box by clicking on it.
- Click the "Delete Student" button to delete the selected student record from the database.
- Upon successful deletion, a message box will confirm the operation, and the entry fields will be cleared.

Student ID:
 Student Name:
 Father's Name:
 Mother's Name:
 Highest Qualification:
 Date of Birth (DOB):
 Address:
 Email:
 Mobile Number:

All Students:

ID: 12, Name: asdf, Father's Name: , Mother's Name: , Highest Qualification: , DOB: , Address: , Email: , Mobile:
ID: asdsad, Name: eqwfw, Father's Name: , Mother's Name: , Highest Qualification: , DOB: , Address: , Email: , Mo
ID: 23msm40001, Name: Ashish, Father's Name: , Mother's Name: , Highest Qualification: , DOB: , Address: , Email:
ID: 23msm40002, Name: Ashish, Father's Name: Kamlesh, Mother's Name: Nirmla, Highest Qualification: BCA, DOI
ID: 123123, Name: dfsf, Father's Name: sdfdsdfs, Mother's Name: dfsfsdf, Highest Qualification: sdfdf, DOB: 4/9/2
ID: 23msm40089, Name: Komalpreet Kaur, Father's Name: Gurbachan Singh, Mother's Name: Pata Nahi, Highest Q
ID: 23MSM40003, Name: Pamanji Jagadeesh, Father's Name: Pamanji, Mother's Name: Pamanji, Highest Qualificat

Success

Student deleted successfully.

OK

Conclusion

The Student Management System provides a user-friendly interface for managing student records efficiently. By leveraging Tkinter for GUI development and MongoDB for data storage, the application offers a reliable solution for educational institutions to streamline their student management processes. With features for adding, viewing, updating, and deleting student records, the system empowers users to maintain accurate and up-to-date student information.