1. admin\_tools.py

import streamlit as st

import pandas as pd

import os

from pymongo import MongoClient

from config import MONGO\_URI

from communication import send\_email\_receipt, send\_sms

from fee\_calculator import generate\_fee\_record

SESSION\_MONTHS = [

    "April", "May", "June", "July", "August", "September",

    "October", "November", "December", "January", "February", "March"

]

def patch\_fee\_ledgers\_streamlit():

    st.subheader("🩺 Fix Fee Ledgers")

    if st.button("🔧 Run Ledger Patch"):

        client = MongoClient(MONGO\_URI)

        db = client["class\_mgmt"]

        students = db["students"]

        fee\_records = db["fee\_records"]

        updated = 0

        for student in students.find({}, {"\_id": 0}):

            sid = student["Student ID"]

            fatherless = student.get("Fatherless", False)

            existing\_months = {r["month"] for r in fee\_records.find({"student\_id": sid})}

            missing\_months = [m for m in SESSION\_MONTHS if m not in existing\_months]

            if missing\_months:

                full\_ledger = generate\_fee\_record(sid, fatherless)

                patch\_docs = [r for r in full\_ledger if r["month"] in missing\_months]

                fee\_records.insert\_many(patch\_docs)

                updated += 1

        if updated:

            st.success(f"✅ Patched fee records for {updated} students.")

        else:

            st.info("All students already have complete fee ledgers.")

def communication\_controls():

    client = MongoClient(MONGO\_URI)

    db = client["class\_mgmt"]

    students = db["students"]

    st.subheader("📬 Guardian Communication Center")

    student\_map = {

        f"{s['Student ID']} – {s['Name']}": s

        for s in students.find({}, {"\_id": 0, "Student ID": 1, "Name": 1, "Guardian Email": 1, "Mobile": 1})

    }

    selected = st.selectbox("Select Student", list(student\_map.keys()))

    student = student\_map[selected]

    student\_id = student["Student ID"]

    email = student.get("Guardian Email", "")

    phone = student.get("Mobile", "")

    month = st.selectbox("Fee Month", SESSION\_MONTHS)

    receipt\_file = f"receipts/FEE2025-{student\_id}-{month}.pdf"

    email\_subject = f"Fee Receipt for {month}"

    email\_body = st.text\_area("📧 Email Message Preview", f"Dear Guardian,\n\nAttached is your official fee receipt for {month}.\n\nBest regards,\nSchool Admin")

    sms\_message = st.text\_area("📱 SMS Message Preview", f"Dear Guardian, fee for {month} has been received. Ref: FEE2025-{student\_id}-{month}.")

    send\_both = st.checkbox("📦 Send Both Email and SMS")

    col1, col2 = st.columns(2)

    with col1:

        if st.button("📧 Send Email"):

            if os.path.exists(receipt\_file):

                sent = send\_email\_receipt(email, email\_subject, email\_body, receipt\_file)

                st.success("✅ Email sent." if sent else "❌ Failed to send email.")

            else:

                st.warning("⚠️ Receipt file not found.")

    with col2:

        if st.button("📱 Send SMS"):

            success = send\_sms(phone, sms\_message)

            st.success("✅ SMS sent." if success else "❌ SMS failed.")

    if send\_both and st.button("🚀 Send Both"):

        sms\_success = send\_sms(phone, sms\_message)

        email\_success = False

        if os.path.exists(receipt\_file):

            email\_success = send\_email\_receipt(email, email\_subject, email\_body, receipt\_file)

        if email\_success and sms\_success:

            st.success("✅ Both Email and SMS sent successfully.")

        elif not email\_success and not sms\_success:

            st.error("❌ Both dispatches failed.")

        else:

            if email\_success:

                st.warning("✅ Email sent, ❌ SMS failed.")

            else:

                st.warning("✅ SMS sent, ❌ Email failed.")

def import\_students\_csv():

    st.subheader("📁 Import Students from CSV")

    uploaded\_file = st.file\_uploader("Upload Student CSV", type=["csv"])

    if uploaded\_file:

        df = pd.read\_csv(uploaded\_file)

        st.write("🔍 Preview of Uploaded Data", df.head())

        if st.checkbox("⚠️ Clear existing MongoDB 'students' collection first"):

            if st.button("🧹 Confirm Clear and Insert"):

                client = MongoClient(MONGO\_URI)

                db = client["class\_mgmt"]

                students\_collection = db["students"]

                students\_collection.delete\_many({})

                students\_collection.insert\_many(df.to\_dict(orient="records"))

                st.success(f"✅ Imported {len(df)} students after clearing existing data.")

        elif st.button("📥 Insert Without Clearing"):

            client = MongoClient(MONGO\_URI)

            db = client["class\_mgmt"]

            students\_collection = db["students"]

            students\_collection.insert\_many(df.to\_dict(orient="records"))

            st.success(f"✅ Imported {len(df)} students into MongoDB.")

def admin\_tools\_panel():

    patch\_fee\_ledgers\_streamlit()

    with st.expander("📬 Open Guardian Communication Center"):

        communication\_controls()

    with st.expander("📁 Import Student Records from CSV"):

        import\_students\_csv()

1. app.py

import streamlit as st

from views.dashboard import student\_dashboard

from views.fee\_view import fee\_view

from views.admin\_tools import admin\_tools\_panel

from login import login\_interface

if "is\_admin" not in st.session\_state:

    st.session\_state["is\_admin"] = False

st.sidebar.image("assets/logo.png", width=150)

st.sidebar.title("Class Manager 🧮")

login\_interface()

st.title("📘 Class Management System")

tabs = ["Dashboard", "Fee Ledger"]

if st.session\_state["is\_admin"]:

    tabs.append("Admin Tools")

selected\_tab = st.sidebar.radio("Navigate", tabs)

if selected\_tab == "Dashboard":

    student\_dashboard()

elif selected\_tab == "Fee Ledger":

    fee\_view()

elif selected\_tab == "Admin Tools" and st.session\_state["is\_admin"]:

    admin\_tools\_panel()

1. communication.py

import smtplib

from email.message import EmailMessage

from twilio.rest import Client

import os

from dotenv import load\_dotenv

load\_dotenv()

# account\_sid = os.getenv("TWILIO\_SID")

# auth\_token = os.getenv("TWILIO\_TOKEN")

# --- EMAIL FUNCTION ---

def send\_email\_receipt(to\_email, subject, body, attachment\_path=None):

    msg = EmailMessage()

    msg["Subject"] = subject

    msg["From"] = "jagdevsinghdosanjh@gmail.com"

    msg["To"] = to\_email

    msg.set\_content(body)

    if attachment\_path and os.path.exists(attachment\_path):

        with open(attachment\_path, "rb") as f:

            file\_data = f.read()

            file\_name = os.path.basename(attachment\_path)

        msg.add\_attachment(file\_data, maintype="application", subtype="pdf", filename=file\_name)

    try:

        with smtplib.SMTP\_SSL("smtp.gmail.com", 465) as smtp:

            smtp.login("jagdevsinghdosanjh@gmail.com", "smartscienceai")  # Replace with secure app password

            smtp.send\_message(msg)

        return True

    except Exception as e:

        print(f"[Email Error] {e}")

        return False

# --- SMS FUNCTION ---

def send\_sms(to\_number, message):

    try:

        account\_sid = os.getenv("TWILIO\_SID")        # Use environment variables for security

        auth\_token = os.getenv("TWILIO\_TOKEN")

        client = Client(account\_sid, auth\_token)

        client.messages.create(

            body=message,

            from\_="+1234567890",  # Replace with your Twilio number

            to=to\_number

        )

        return True

    except Exception as e:

        print(f"[SMS Error] {e}")

        return False

1. config.py

# MONGO\_URI = "mongodb+srv://<username>:<password>@cluster.mongodb.net/class\_mgmt"

MONGO\_URI = "mongodb://localhost:27017/class\_mgmt"

#mongodb+srv://jagdevsinghdosanjh:ndxjuLnqz4oCcg8z@cluster0.3xnlzlw.mongodb.net/

#MONGO\_URI = "mongodb+srv://jagdevsinghdosanjh:ndxjuLnqz4oCcg8z@cluster0.3xnlzlw.mongodb.net/class\_mgmt?retryWrites=true&w=majority"

1. dashboard.py

from pymongo import MongoClient

import pandas as pd

from config import MONGO\_URI

# Initialize MongoDB client and database

client = MongoClient(MONGO\_URI)

db = client["class\_mgmt"]

students\_collection = db["students"]

def load\_students\_from\_csv(csv\_path="data/student.csv"):

    df = pd.read\_csv(csv\_path)

    records = df.to\_dict(orient="records")

    students\_collection.insert\_many(records)

    return len(records)

def fetch\_all\_students():

    return list(students\_collection.find({}, {"\_id": 0}))

def find\_students\_by\_field(field, value):

    query = {field: {"$regex": value, "$options": "i"}}

    return list(students\_collection.find(query, {"\_id": 0}))

def get\_birthdays\_by\_month(month\_str):

    return list(students\_collection.find({"DOB": {"$regex": f"-{month\_str}-", "$options": "i"}}, {"\_id": 0}))

1. fee\_calculator.py

from datetime import datetime

# Fee breakdown for non-fatherless students

FEE\_STRUCTURE = {

    "Admission Fee": 0,

    "Tuition Fee": 0,

    "Absentee Fine": 0,

    "Late Fee Fine": 0,

    "Amalgamated Fund": 20,

    "PTA Fund": 15,

    "Sports Fund": 15,

    "Other": 5,

    "Continuation Fee": 200

}

TOTAL\_MONTHLY\_FEE = sum(FEE\_STRUCTURE.values())

# Academic session months

SESSION\_MONTHS = [

    'April', 'May', 'June', 'July', 'August', 'September',

    'October', 'November', 'December',

    'January', 'February', 'March'

]

def calculate\_monthly\_fee(fatherless: bool) -> int:

    return 0 if fatherless else TOTAL\_MONTHLY\_FEE

def generate\_fee\_record(student\_id: int, fatherless: bool):

    fee\_records = []

    for i, month in enumerate(SESSION\_MONTHS):

        # Split session year: April–Dec is start year, Jan–March is next year

        year = 2025 if i < 9 else 2026

        fee\_due = calculate\_monthly\_fee(fatherless)

        fee\_records.append({

            "student\_id": student\_id,

            "month": month,

            "year": str(year),

            "fee\_due": fee\_due,

            "paid": False

        })

    return fee\_records

1. fee\_patch\_tool.py

from pymongo import MongoClient

from config import MONGO\_URI

from fee\_calculator import generate\_fee\_record, SESSION\_MONTHS

client = MongoClient(MONGO\_URI)

db = client["class\_mgmt"]

students = db["students"]

fee\_records = db["fee\_records"]

def patch\_fee\_ledgers():

    print("🧪 Checking and patching fee records...")

    updated\_students = []

    for student in students.find({}, {"\_id": 0}):

        sid = student["Student ID"]

        fatherless = student.get("Fatherless", False)

        existing\_months = {r["month"] for r in fee\_records.find({"student\_id": sid})}

        missing\_months = [m for m in SESSION\_MONTHS if m not in existing\_months]

        if missing\_months:

            print(f"🔧 Student ID {sid} is missing months: {missing\_months}")

            full\_ledger = generate\_fee\_record(sid, fatherless)

            patch\_docs = [r for r in full\_ledger if r["month"] in missing\_months]

            fee\_records.insert\_many(patch\_docs)

            updated\_students.append(sid)

    if updated\_students:

        print(f"\n✅ Patched fee records for {len(updated\_students)} students.")

    else:

        print("✅ All students have complete fee ledgers.")

if \_\_name\_\_ == "\_\_main\_\_":

    patch\_fee\_ledgers()

1. fee\_view.py

import streamlit as st

import logging

from database import students\_collection

from fee\_calculator import generate\_fee\_record

# MongoDB collection for fee records

fee\_collection = students\_collection.database["fee\_records"]

# Configure audit logging

logging.basicConfig(filename="fee\_updates.log", level=logging.INFO)

def create\_fee\_ledger(student):

    """Generate fee ledger and insert records."""

    student\_id = student["Student ID"]

    fatherless = student.get("Fatherless", False)

    ledger = generate\_fee\_record(student\_id, fatherless)

    fee\_collection.insert\_many(ledger)

def fetch\_fee\_records(student\_id):

    """Retrieve fee records for a student."""

    return list(fee\_collection.find({"student\_id": student\_id}, {"\_id": 0}))

def update\_payment\_status(student\_name, student\_id, selected\_months, unpaid\_months):

    """Update paid status and log each update."""

    for record in unpaid\_months:

        label = f"{record['month']} {record['year']}"

        if label in selected\_months:

            fee\_collection.update\_one(

                {"student\_id": student\_id, "month": record["month"], "year": record["year"]},

                {"$set": {"paid": True}}

            )

            logging.info(f"{student\_name} | Paid: {label}")

def fee\_view():

    st.title("💰 Fee Ledger Viewer")

    # Load all students

    students = list(students\_collection.find({}, {"\_id": 0}))

    student\_names = [s["Name"] for s in students]

    selected\_name = st.selectbox("Select Student", student\_names)

    student = next((s for s in students if s["Name"] == selected\_name), None)

    if not student:

        st.warning("Student not found.")

        return

    student\_id = student["Student ID"]

    existing = fee\_collection.count\_documents({"student\_id": student\_id})

    if existing == 0:

        create\_fee\_ledger(student)

    records = fetch\_fee\_records(student\_id)

    st.subheader(f"📋 Fee Ledger for {student['Name']}")

    for record in records:

        status = "✅ Paid" if record["paid"] else "❌ Unpaid"

        st.markdown(f"- {record['month']} {record['year']}: ₹{record['fee\_due']} {status}")

    unpaid\_months = [r for r in records if not r["paid"]]

    if unpaid\_months:

        unpaid\_labels = [f"{r['month']} {r['year']}" for r in unpaid\_months]

        selected = st.multiselect("Mark Paid Months", unpaid\_labels)

        if st.button("Update Payment Status"):

            update\_payment\_status(student["Name"], student\_id, selected, unpaid\_months)

            st.success("✅ Payment status updated!")

1. login.py

import streamlit as st

import hashlib

# User database simulation (use MongoDB later if needed)

USER\_CREDENTIALS = {

    "admin": hashlib.sha256("password123".encode()).hexdigest()

}

def login\_interface():

    st.sidebar.subheader("🔐 Admin Login")

    username = st.sidebar.text\_input("Username")

    password = st.sidebar.text\_input("Password", type="password")

    if st.sidebar.button("Login"):

        hashed = hashlib.sha256(password.encode()).hexdigest()

        if USER\_CREDENTIALS.get(username) == hashed:

            st.session\_state["is\_admin"] = True

            st.success("Welcome, Admin 👋")

        else:

            st.error("Invalid credentials")

# Initialize session state (can also be done in app.py once)

if "is\_admin" not in st.session\_state:

    st.session\_state["is\_admin"] = False

1. schema\_validator.py

from pymongo import MongoClient

from config import MONGO\_URI

from fee\_calculator import SESSION\_MONTHS

client = MongoClient(MONGO\_URI)

db = client["class\_mgmt"]

students = db["students"]

fee\_records = db["fee\_records"]

REQUIRED\_STUDENT\_FIELDS = [

    "R.No", "Student ID", "Name", "FatherName", "MotherName",

    "DOB", "Gender", "Class", "Section", "ContactNo", "Fatherless"

]

def validate\_student\_schema():

    print("🔍 Validating student documents...")

    all\_students = students.find()

    missing\_fields\_report = []

    for student in all\_students:

        missing = [field for field in REQUIRED\_STUDENT\_FIELDS if field not in student]

        if missing:

            missing\_fields\_report.append({

                "Student ID": student.get("Student ID", "Unknown"),

                "Missing Fields": missing

            })

    if missing\_fields\_report:

        print("❗ Students with missing fields:")

        for entry in missing\_fields\_report:

            print(f"- ID {entry['Student ID']}: Missing {entry['Missing Fields']}")

    else:

        print("✅ All students have complete schema.")

def validate\_fee\_records():

    print("\n📊 Validating fee records...")

    all\_students = students.find()

    issues\_found = []

    for student in all\_students:

        sid = student["Student ID"]

        fee\_docs = list(fee\_records.find({"student\_id": sid}))

        if len(fee\_docs) != 12:

            issues\_found.append(f"Student ID {sid} has {len(fee\_docs)} fee records (expected 12).")

        # Check month consistency

        recorded\_months = {doc["month"] for doc in fee\_docs}

        missing\_months = set(SESSION\_MONTHS) - recorded\_months

        if missing\_months:

            issues\_found.append(f"Student ID {sid} is missing months: {sorted(list(missing\_months))}")

    if issues\_found:

        print("❗ Fee record issues found:")

        for issue in issues\_found:

            print(f"- {issue}")

    else:

        print("✅ All students have complete fee records for April–March.")

if \_\_name\_\_ == "\_\_main\_\_":

    validate\_student\_schema()

    validate\_fee\_records()

1. requirements.txt

streamlit==1.32.2         # For building interactive UI

pymongo==4.6.1            # MongoDB database operations

pandas==2.2.2             # Data handling and exports

bcrypt==4.1.3             # Secure password hashing

fpdf==1.7.2               # PDF receipt generation

openpyxl==3.1.2           # Excel export support

xlsxwriter==3.1.9         # Alternative Excel writer with styling

python-dateutil==2.9.0    # Date parsing and validation

python-dotenv==1.0.0

twilio==9.6.5

1. .env

TWILIO\_SID=myTwilioSID

TWILIO\_TOKEN=myTwilioToken

1. .env.example

# Twilio Credentials for SMS Communication

TWILIO\_SID=TwilioSID\_thatbelongstome

TWILIO\_TOKEN=myTwilioToken

# Optional: SMTP Email Settings (if adding secure email dispatch)

EMAIL\_ADDRESS=jagdevsinghdosanjh@gmail.com

EMAIL\_PASSWORD=MyGmailAppPassword

1. studentdata.csc

R.No,Student ID,Name,FatherName,MotherName,DOB,Gender,Class,**Section,**ContactNo

1,8792461,AJANBIR SINGH,DAVINDER SINGH,MANINDER KAUR,10-Oct-2010,Male,10th,**A,**7508144649

2,9012827,ANMOLPREET KAUR,SARWAN SINGH,RANJIT KAUR,29-Jan-2010,Female,10th,**A,**8198892150

3,7069347,ARMAANDEEP SINGH,JASWANT SINGH,HARJINDER KAUR,27-Dec-2010,Male,10th,**A,**9781645863

4,8614037,ARMANDEEP KAUR,DALBIR SINGH,RAMANDEEP KAUR,01-Sep-2010,Female,10th,**A,**9855489155

5,8790511,ARMANDEEP SINGH,NIRMAL SINGH,BALJIT KAUR,21-Oct-2010,Male,10th,**A,**7070071702

6,13890949,BALRAJ SINGH,BIKRAMJIT SINGH,LAKWINDER KAUR,09-Mar-2012,Male,10th,**A,**9915843409

7,15773856,BIKRAMJIT SINGH,JANTA,MANROOP,23-Apr-2010,Male,10th,**A,**7087821776

8,7077019,DILPREET KAUR,NARINDER SINGH,SHARANJIT KAUR,10-Sep-2009,Female,10th,**A,**9779286111

9,7080980,GURPREET KAUR,NARINDER SINGH,KULWANT KAUR,16-Aug-2010,Female,10th,**A,**8146531652

10,3958507,JARMANJEET SINGH,SAHIB SINGH,SUKHWINDER KAUR,02-Oct-2008,Male,10th,**A,**9876027537

11,8789695,JASHANPREET SINGH,SATNAM SINGH,MADANPREET KAUR,31-May-2010,Male,10th,**A,**9981476510

12,9062017,JASPREET KAUR,GURJIT SINGH,LAKHWINDER KAUR,05-Feb-2011,Female,10th,**A,**9855764423

13,7069339,JIVAN SINGH,GURDEEP SINGH,SARABJIT KAUR,27-Jul-2009,Male,10th,**A,**9814543068

14,8790429,KARANDEEP SINGH,BALJIT SINGH,JASPAL KAUR,15-Jun-2010,Male,10th,**A,**9876321762

15,8790588,KARANPREET SINGH,RAJPAL SINGH,PARAMJIT KAUR,11-May-2010,Male,10th,**A,**7710445468

16,9198115,KHUSHPREET KAUR,NISHAN SINGH,SIMRANJIT KAUR,16-Aug-2011,Female,10th,**A,**9878372657

17,8792320,KOMALPREET KAUR,DILBAG SINGH,DALJIT KAUR,01-Mar-2011,Female,10th,**A,**9501431895

18,8790676,KULJIT KAUR,MANGAL SINGH,JASPINDER KAUR,26-Jan-2011,Female,10th,**A,**7802961791

19,8790345,LOVEJOT SINGH,SADHA SINGH,JASWANT KAUR,22-Mar-2011,Male,10th,**A,**8968538220

20,8806313,LOVEJOT SINGH,JASWANT SINGH,HARJINDER KAUR,08-Dec-2011,Male,10th,**A,**8146549029

21,8797009,MANJOT KAUR,GURPEET SINGH,RAJBIR KAUR,21-Sep-2011,Female,10th,**A,**8284063293

22,7080966,MANPREET KAUR,NARINDER SINGH,KULWANT KAUR,02-Feb-2008,Female,10th,**A,**8146531652

23,8792146,MUSKANDEEP KAUR,MADHA SINGH,JASBIR KAUR,30-Jul-2010,Female,10th,**A,**8264584731

24,8797216,MUSKANPREET KAUR,BHAGWANT SINGH,PALWINDER KAUR,01-Nov-2010,Female,10th,**A,**8146531652

25,8797160,NEELAM KAUR,JASWANT SINGH,MANPREET KAUR,09-Mar-2012,Female,10th,**A,**8146531652

26,8806296,PAWANDEEP KAUR,GURDEEP SINGH,SARBJIT KAUR,02-Oct-2011,Female,10th,**A,**8146549029

27,8797236,RAJVEER SINGH,KULDEEP SINGH,MANPREET KAUR,23-Dec-2010,Male,10th,**A,**8146531652

28,9777058,RAJWINDER KAUR,MAJOR SINGH,PREET KAUR,25-Jun-2011,Female,10th,**A,**9814043592

29,8790228,SAHILPREET SINGH,GURBINDER SINGH,SARBJIT KAUR,10-Feb-2011,Male,10th,**A,**8427168615

30,8792207,SANDEEP KAUR,GURMEET SINGH,SONIA,02-Mar-2011,Female,10th,**A,**9878191528

31,8835723,SANDEEP SINGH,JASPAL SINGH,GURPREET KAUR,23-Sep-2010,Male,10th,**A,**8544966373

32,8797179,SARTAJ SINGH,MANGAL SINGH,RAJWINDER KAUR,01-Dec-2011,Male,10th,**A,**8146531652

33,9062159,SATBIR KAUR,HARJINDER SINGH,KARMJIT KAUR,30-Jul-2010,Female,10th,**A,**9876177149

34,10352570,SIMARJEET KAUR,AMARJIT SINGH,RANJIT KAUR,05-Oct-2007,Female,10th,**A,**9501048483

35,10118973,SIMRANJEET KAUR,PARVINDER SINGH,RANJIT KAUR,23-Dec-2010,Female,10th,**A,**9592864408

36,8814734,VANSHDEEP SINGH,TARANJIT SINGH,KULWINDER KAUR,26-Jan-2010,Male,10th,**A,**7347471523

37,7407917,VANSHPREET SINGH,GURBINDER SINGH,SARBJIT KAUR,03-Jan-2010,Male,10th,**A,**9779844896

Directory Structure for my VS Code Project

