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()
```

2. 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()
```

3. 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
```

4. config.py

```
# MONGO_URI =
"mongodb+srv://<username>:<password>@cluster.mongodb.net/class_
mgmt"

MONGO_URI = "mongodb://localhost:27017/class_mgmt"

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

#MONGO_URI =
"mongodb+srv://jagdevsinghdosanjh:ndxjuLnqz4oCcg8z@cluster0.3xnlzl
w.mongodb.net/class_mgmt?retryWrites=true&w=majority"
```

5. 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}))
```

6. 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
```

7. 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()
```

8. 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!")
```

9. 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
```

10. 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()
```

11. 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
```

12. .env

```
TWILIO_SID=myTwilioSID
TWILIO_TOKEN=myTwilioToken
```

13. .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
```

14. 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

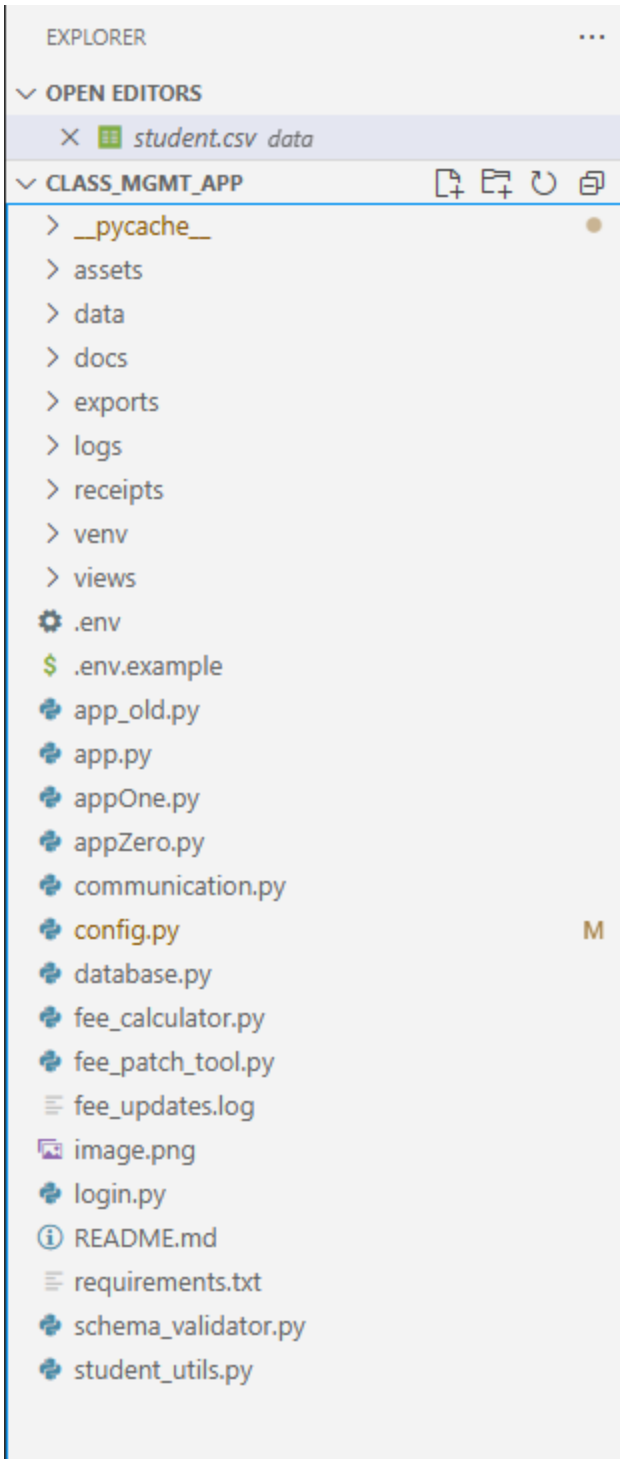
34,10352570,SIMARJEET KAUR,AMARJIT SINGH,RANJIT KAUR,05-Oct-2007,Female,10th,**A,**9501048483

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


EXPLORER
OPEN EDITORS
student.csv data
CLASS_MGMT_APP
__pycache__
assets
data
docs
exports
logs
receipts
venv
views
.env
.env.example
app_old.py
app.py
appOne.py
appZero.py
communication.py
config.py
M
database.py
fee_calculator.py
fee_patch_tool.py
fee_updates.log
image.png
login.py
README.md
requirements.txt
schema_validator.py
student_utils.py