# db\_handler.py

import sqlite3

DB\_NAME = 'hostel.db'

def connect():

return sqlite3.connect(DB\_NAME)

def create\_tables():

conn = connect()

cursor = conn.cursor()

# Students Table

cursor.execute('''

CREATE TABLE IF NOT EXISTS students (

student\_id TEXT PRIMARY KEY,

name TEXT NOT NULL,

age INTEGER,

course TEXT,

room\_number INTEGER,

FOREIGN KEY (room\_number) REFERENCES rooms(room\_number)

)

''')

# Rooms Table

cursor.execute('''

CREATE TABLE IF NOT EXISTS rooms (

room\_number INTEGER PRIMARY KEY,

capacity INTEGER,

occupants INTEGER

)

''')

# Payments Table

cursor.execute('''

CREATE TABLE IF NOT EXISTS payments (

payment\_id INTEGER PRIMARY KEY AUTOINCREMENT,

student\_id TEXT,

amount REAL,

date TEXT,

FOREIGN KEY (student\_id) REFERENCES students(student\_id)

)

''')

# Admin login (only one warden user)

cursor.execute('''

CREATE TABLE IF NOT EXISTS admin (

username TEXT PRIMARY KEY,

password TEXT

)

''')

# Insert default warden if not exist

cursor.execute("INSERT OR IGNORE INTO admin (username, password) VALUES (?, ?)", ('warden', 'password'))

# Insert rooms if not exist

for room in range(1, 101):

cursor.execute("INSERT OR IGNORE INTO rooms (room\_number, capacity, occupants) VALUES (?, ?, ?)", (room, 2, 0))

conn.commit()

conn.close()

def authenticate(username, password):

conn = connect()

cursor = conn.cursor()

cursor.execute("SELECT \* FROM admin WHERE username=? AND password=?", (username, password))

result = cursor.fetchone()

conn.close()

return result

def add\_student(student\_id, name, age, course):

conn = connect()

cursor = conn.cursor()

# Find available room

cursor.execute("SELECT room\_number FROM rooms WHERE occupants < capacity LIMIT 1")

room = cursor.fetchone()

if room:

room\_number = room[0]

cursor.execute("INSERT INTO students (student\_id, name, age, course, room\_number) VALUES (?, ?, ?, ?, ?)",

(student\_id, name, age, course, room\_number))

cursor.execute("UPDATE rooms SET occupants = occupants + 1 WHERE room\_number = ?", (room\_number,))

conn.commit()

conn.close()

return True, f"Assigned Room {room\_number}"

else:

conn.close()

return False, "No rooms available!"

def get\_students():

conn = connect()

cursor = conn.cursor()

cursor.execute("SELECT \* FROM students")

data = cursor.fetchall()

conn.close()

return data

def search\_students(query):

conn = connect()

cursor = conn.cursor()

cursor.execute("SELECT \* FROM students WHERE student\_id LIKE ? OR name LIKE ?", (f"%{query}%", f"%{query}%"))

data = cursor.fetchall()

conn.close()

return data

def delete\_student(student\_id):

conn = connect()

cursor = conn.cursor()

cursor.execute("SELECT room\_number FROM students WHERE student\_id=?", (student\_id,))

room = cursor.fetchone()

if room:

room\_number = room[0]

cursor.execute("UPDATE rooms SET occupants = occupants - 1 WHERE room\_number = ?", (room\_number,))

cursor.execute("DELETE FROM students WHERE student\_id=?", (student\_id,))

conn.commit()

conn.close()

def add\_payment(student\_id, amount, date):

conn = connect()

cursor = conn.cursor()

cursor.execute("INSERT INTO payments (student\_id, amount, date) VALUES (?, ?, ?)", (student\_id, amount, date))

conn.commit()

conn.close()

def get\_payments():

conn = connect()

cursor = conn.cursor()

cursor.execute("SELECT \* FROM payments")

data = cursor.fetchall()

conn.close()

return data

def get\_rooms():

conn = connect()

cursor = conn.cursor()

cursor.execute("SELECT \* FROM rooms")

data = cursor.fetchall()

conn.close()

return data