Advanced Python Project: Flask API, AWS Integration, and Machine Learning

This document contains an advanced Python project that demonstrates enterprise-grade skills in Flask API development, AWS S3 integration, machine learning, multi-threading, database operations, and error handling. This project follows best practices and is designed to impress MAANG recruiters.

```
import threading
import time
import logging
import requests
import sqlite3
import json
import os
import boto3
from flask import Flask, jsonify, request
from sklearn.linear_model import LinearRegression
import numpy as np
import pandas as pd
from concurrent.futures import ThreadPoolExecutor
import unittest

# Configure logging
logging.basicConfig(filename="app.log", level=logging.INFO,
                    format="%(asctime)s - %(levelname)s - %(message)s")

# Database Manager
class DatabaseManager:
    def __init__(self, db_name="app_data.db"):
        self.conn = sqlite3.connect(db_name, check_same_thread=False)
        self.cursor = self.conn.cursor()
        self.create_table()

    def create_table(self):
        self.cursor.execute("""
            CREATE TABLE IF NOT EXISTS users (
                id INTEGER PRIMARY KEY AUTOINCREMENT,
```

```
            name TEXT NOT NULL,
            email TEXT UNIQUE NOT NULL
        )
    """)
    self.conn.commit()

  def insert_user(self, name, email):
    try:
      self.cursor.execute("INSERT INTO users (name, email) VALUES (?, ?)", (name,
email))
      self.conn.commit()
      logging.info(f"User {name} added successfully.")
    except sqlite3.IntegrityError:
      logging.warning("Duplicate email detected.")

  def fetch_users(self):
    self.cursor.execute("SELECT * FROM users")
    return self.cursor.fetchall()

  def close(self):
    self.conn.close()

db = DatabaseManager()

# Flask API
app = Flask(__name__)

@app.route('/users', methods=['GET'])
def get_users():
  users = db.fetch_users()
  return jsonify(users)

@app.route('/add_user', methods=['POST'])
def add_user():
  data = request.get_json()
  db.insert_user(data['name'], data['email'])
  return jsonify({"message": "User added successfully!"})

# AWS S3 Integration
AWS_ACCESS_KEY = "YOUR_ACCESS_KEY"
```

```
AWS_SECRET_KEY = "YOUR_SECRET_KEY"
AWS_BUCKET_NAME = "your-s3-bucket"

s3_client = boto3.client(
    "s3",
    aws_access_key_id=AWS_ACCESS_KEY,
    aws_secret_access_key=AWS_SECRET_KEY
)

def upload_file_to_s3(file_path, bucket, s3_filename):
    try:
        s3_client.upload_file(file_path, bucket, s3_filename)
        logging.info(f"Uploaded {file_path} to S3 bucket {bucket} as {s3_filename}")
        return f"https://{bucket}.s3.amazonaws.com/{s3_filename}"
    except Exception as e:
        logging.error(f"S3 Upload Error: {str(e)}")
        return None

# ML Model - Predicting House Prices
def train_ml_model():
    data = pd.DataFrame({
        "size": [750, 800, 850, 900, 1000, 1100],
        "price": [150000, 160000, 170000, 180000, 200000, 220000]
    })

    X = data["size"].values.reshape(-1, 1)
    y = data["price"].values

    model = LinearRegression()
    model.fit(X, y)

    return model

model = train_ml_model()

@app.route('/predict_price', methods=['POST'])
def predict_price():
    data = request.get_json()
    size = np.array([[data["size"]]])
    price = model.predict(size)[0]
```

```
    return jsonify({"predicted_price": round(price, 2)})

if __name__ == "__main__":
    logging.info("Application Started")
    unittest.main(exit=False)
    db.close()
    logging.info("Application Finished")
```