```
# capstone_project.py
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from datetime import datetime, timedelta

# ------------------------------------------------------------
# 1) SAMPLE DATA GENERATION (AUTO)
# ------------------------------------------------------------
def generate_sample_data():
os.makedirs("data", exist_ok=True)

def create_csv(building_name):
start = datetime.now() - timedelta(days=30)
date_range = pd.date_range(start=start, periods=30*24, freq="H")
hours = date_range.hour
base = 5 + (np.sin(hours/24 * 2*np.pi) + 1) * 3
noise = np.random.normal(0, 0.8, len(date_range))
kwh = np.maximum(0.1, base + noise + np.random.rand()*2)

df = pd.DataFrame({
timestamp: date_range,
kwh: np.round(kwh, 3)
})
df.to_csv(f"data/{building_name}.csv", index=False)

create_csv("engineering_block")
create_csv("library")
create_csv("hostel_A")
print("Sample Data Generated (3 buildings, 30 days hourly).")


# ------------------------------------------------------------
# 2) INGESTION
# ------------------------------------------------------------
def load_all_data(data_path="data/"):
combined = []
errors = []

if not os.path.exists(data_path):
os.makedirs(data_path)
```

```
for fname in os.listdir(data_path):
    if not fname.endswith(".csv"):
        continue
    try:
        df = pd.read_csv(os.path.join(data_path, fname), parse_dates=["timestamp"])
        if "kwh" not in df.columns:
            errors.append(f"{fname}: missing kwh column")
            continue
        df["building"] = fname.replace(".csv", "")
        combined.append(df)
    except Exception as e:
        errors.append(str(e))

if not combined:
    return pd.DataFrame(), errors

final_df = pd.concat(combined, ignore_index=True)
final_df = final_df.sort_values("timestamp")
return final_df, errors


# ------------------------------------------------------------
# 3) AGGREGATION ENGINE
# ------------------------------------------------------------
def daily_totals(df):
    return df.groupby(df["timestamp"].dt.date)["kwh"].sum()

def weekly_totals(df):
    return df.groupby(df["timestamp"].dt.to_period("W"))["kwh"].sum()

def building_summary(df):
    return df.groupby("building")["kwh"].agg(["mean", "min", "max", "sum"]).round(3)


# ------------------------------------------------------------
# 4) VISUALIZATION
# ------------------------------------------------------------
def create_dashboard(daily, weekly, df):
    os.makedirs("output", exist_ok=True)

    fig, axes = plt.subplots(3, 1, figsize=(10, 13), constrained_layout=True)

    # Daily line chart
```

```
axes[0].plot(daily.index, daily.values)
axes[0].set_title("Daily Energy Consumption")
axes[0].set_ylabel("kWh")

# Weekly bar chart
axes[1].bar(weekly.index.astype(str), weekly.values)
axes[1].set_title("Weekly Consumption")
axes[1].tick_params(axis="x", rotation=45)

# Scatter plot by building
for b, group in df.groupby("building"):
axes[2].scatter(group["timestamp"], group["kwh"], label=b, s=10)
axes[2].set_title("Hourly Readings")
axes[2].legend()

plt.savefig("output/dashboard.png")
plt.close()


# ------------------------------------------------------------
# 5) SUMMARY TXT & CSV EXPORT
# ------------------------------------------------------------
def export_outputs(df, bsum):
os.makedirs("output", exist_ok=True)

df.to_csv("output/cleaned_data.csv", index=False)
bsum.to_csv("output/building_summary.csv")

with open("output/summary.txt", "w") as f:
f.write("Campus Energy Report\n")
f.write("====================\n\n")
f.write(f"Total Campus Consumption: {df['kwh'].sum():.2f} kWh\n")
high = bsum["sum"].idxmax()
f.write(f"Highest Consuming Building: {high} ({bsum.loc[high,'sum']:.2f} kWh)\n")


# ------------------------------------------------------------
# MAIN PIPELINE
# ------------------------------------------------------------
def run_pipeline():
print("\n>>> Generating Sample Data...")
generate_sample_data()
```

```
print("\n>>> Loading all CSV data...")
df, errors = load_all_data()

print("Rows Loaded:", len(df))
if errors:
print("Errors:", errors)

print("\n>>> Calculating Aggregates...")
d = daily_totals(df)
w = weekly_totals(df)
b = building_summary(df)

print("\n>>> Creating Dashboard...")
create_dashboard(d, w, df)

print("\n>>> Exporting CSVs and Summary...")
export_outputs(df, b)

print("\nCompleted Successfully! Check 'output/' folder.\n")


# ------------------------------------------------------------
# Execute
# ------------------------------------------------------------
if _name_ == "_main_":
run_pipeline()
```