

Overview — Problem & Solution

Problem: many clinics/SME hospitals keep patient and operational data in spreadsheets or fragmented systems, so they can't see real-time KPIs (admissions, bed occupancy, average length of stay), predict readmissions, or detect resource shortages.

Solution (MVP): a web dashboard that ingests CSV/DB data, shows interactive patient & operational KPIs, predicts 30-day readmission risk for admitted patients, and raises simple alerts (low beds, high risk patients). Easy to demo and customizable per client.

What the MVP will do (core features)

1. Data ingestion (CSV / Google Sheets / small SQL DB)
 2. Dashboards:
 - Daily admissions, monthly revenue, avg length of stay (ALOS)
 - Bed occupancy % and ward-level view
 - Top diagnoses and top services
 3. Predictive model: **30-day readmission risk** (binary) with a risk score per patient
 4. Alerts:
 - Highlight patients with high readmission risk
 - Alert when bed occupancy > threshold
 5. Export reports (CSV/PDF) and simple “predict” form for single patients
 6. Admin page to upload data and retrain model
-

Data & Schema (sample)

Build demo CSV files or use a small SQLite DB.

patients.csv

- patient_id, first_name, last_name, dob, gender, phone

admissions.csv

- admission_id, patient_id, admit_date, discharge_date (nullable), diagnosis, ward, attending_physician, admission_type (elective/emergency), total_charges

vitals.csv (optional)

- admission_id, timestamp, heart_rate, bp_systolic, bp_diastolic, temp

labs.csv (optional)

- admission_id, lab_date, test_name, value

appointments.csv

- appointment_id, patient_id, date, status

sample column (admissions.csv):

A001, P123, 2024-07-01, 2024-07-05, Pneumonia, Ward A, Dr. Mwangi, emergency, 4500.00

Tech stack (recommended MVP)

- **Language:** Python 3.10+
 - **Data / ML:** pandas, numpy, scikit-learn, joblib
 - **Web UI / Dashboard:** Streamlit (fast) OR Flask + React (more work)
 - **Visualization:** plotly.express or matplotlib
 - **DB (optional):** SQLite for demo, PostgreSQL for production
 - **Alerts:** SMTP (email) or Twilio / WhatsApp Cloud API for WhatsApp alerts
 - **Deployment:** Streamlit Cloud / Render / Heroku / Vercel (frontend)
 - **Versioning / Packaging:** git, Docker (optional)
-

Step-by-step implementation

Phase 0 — Setup (minutes)

```
bash
CopyEdit
python -m venv venv
source venv/bin/activate          # Windows: venv\Scripts\activate
pip install pandas numpy scikit-learn streamlit plotly joblib sqlalchemy
```

Phase 1 — Prepare sample data (1 day)

- Create patients.csv, admissions.csv, vitals.csv with realistic rows.
- Make sure dates are ISO YYYY-MM-DD.

Phase 2 — Data cleaning & feature engineering (1–2 days)

Create a Python script `process_data.py` to:

- Parse dates, compute `length_of_stay = discharge_date - admit_date (days)`
- Compute past admissions count per patient
- Encode categorical fields (ward, diagnosis simplified)
- Create target `readmitted_30d` (for historical data: patient readmitted within 30 days after discharge)

Example snippet:

```
python
CopyEdit
import pandas as pd
from datetime import timedelta

adm = pd.read_csv('admissions.csv',
parse_dates=['admit_date', 'discharge_date'])
# length of stay
adm['los'] = (adm['discharge_date'] -
adm['admit_date']).dt.days.clip(lower=0)
# previous admissions count
adm = adm.sort_values(['patient_id', 'admit_date'])
adm['prev_adm_count'] = adm.groupby('patient_id').cumcount()
# compute readmission within 30 days
adm['next_admit_date'] = adm.groupby('patient_id')['admit_date'].shift(-1)
adm['readmit_30d'] = ((adm['next_admit_date'] -
adm['discharge_date']).dt.days <= 30).astype(int)
adm.to_csv('admissions_processed.csv', index=False)
```

Phase 3 — Train a simple model (2 days)

Use RandomForest or LogisticRegression to predict `readmit_30d`. Keep it explainable.

```
python
CopyEdit
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import roc_auc_score, classification_report
import joblib
import pandas as pd

df = pd.read_csv('admissions_processed.csv')
features = ['los', 'prev_adm_count'] # add encoded features as needed
X = df[features].fillna(0)
y = df['readmit_30d']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
clf = RandomForestClassifier(n_estimators=100, random_state=42)
clf.fit(X_train, y_train)

y_pred = clf.predict_proba(X_test)[:,1]
print("AUC:", roc_auc_score(y_test, y_pred))
```

```
joblib.dump(clf, 'model_readmit.joblib')
```

Note: Track AUC, precision/recall, and consider calibration if probabilities used.

Phase 4 — Build Streamlit dashboard (2–4 days)

A simple Streamlit app `app.py` that:

- Lets admin upload CSVs
- Shows KPI cards: total admissions, avg LOS, bed occupancy
- Displays charts with plotly (time series of admissions)
- Provides a “Predict” form: input LOS, prev_adm_count → show risk score from model
- Table of patients flagged as high-risk

Example skeleton:

```
python
CopyEdit
import streamlit as st
import pandas as pd
import joblib
import plotly.express as px

st.title("SmartBI — Patient Analytics")

uploaded = st.file_uploader("Upload admissions CSV", type="csv")
if uploaded:
    adm = pd.read_csv(uploaded, parse_dates=['admit_date', 'discharge_date'])
    adm['los'] = (adm['discharge_date'] -
adm['admit_date']).dt.days.clip(lower=0)
    st.metric("Total Admissions", len(adm))
    st.metric("Avg LOS", round(adm['los'].mean(), 1))
    fig =
px.line(adm.groupby(adm['admit_date'].dt.to_period('M')).size().reset_index(n
ame='count'), x='admit_date', y='count')
    st.plotly_chart(fig, use_container_width=True)

# load model
model = joblib.load('model_readmit.joblib')
st.subheader("Predict readmission risk")
los = st.number_input("Length of stay (days)", min_value=0, value=3)
prev = st.number_input("Previous admissions", min_value=0, value=0)
if st.button("Predict Risk"):
    prob = model.predict_proba([[los, prev]])[0,1]
    st.write(f"Readmission risk (30d): {prob:.2%}")
```

Phase 5 — Alerts (1 day)

- Create logic to flag patients with predicted risk > threshold (e.g., 0.6).
- Send a notification email via SMTP or send an HTTP request to Twilio/WhatsApp Cloud API for alerts.

- Example: using Python `smtpplib` for email.

Phase 6 — Validation & clinician feedback (ongoing)

- Share demo with a clinician; collect feedback and adjust features.
- Retrain model with more data and possible additional features (labs, comorbidities).

Phase 7 — Deployment (1–2 days)

- Deploy Streamlit app to **Streamlit Cloud** (fast), or package as Docker and deploy on **Render** / **Heroku**.
 - If connecting to hospital systems later, use **FHIR** (standards) for EHR integration.
-

Evaluation & Metrics

- **For the predictive model:** use AUC, precision at high recall (you may prefer recall if missing high-risk patients is costly), and confusion matrix.
 - **System metrics:** dashboard load time, model inference time, correctness of KPI calculations.
-

Privacy & Security (critical for healthcare)

- **Always** handle PHI carefully. For demos use anonymized data (remove names, phone numbers).
 - Use **HTTPS**, authentication (simple admin login), role-based access.
 - Store minimal data, encrypt backups, implement audit logs.
 - Check local data protection laws (e.g., Kenya Data Protection Act) — get patient consent when needed.
-

Simple MVP Timeline (2–3 weeks)

- Days 1–3: Data & schema + sample data
- Days 4–6: Data cleaning & feature engineering
- Days 7–9: Train model and evaluate
- Days 10–13: Build Streamlit dashboard + prediction form
- Days 14–16: Alerts integration, testing with sample data

- Days 17–20: Deploy + prepare demo materials

(You can compress to 10 days for a lean demo.)

Next steps I can do for you (pick one)

- Provide the **complete starter code** (data processing + training + Streamlit app) that you can run locally.
- Prepare a **demo dataset** (synthesized) ready to upload.
- Help you **design the UI** (mockups) or prepare a pitch/demo script for clinics.

Which would you like me to prepare now? I can start by generating the **starter code + sample dataset** for immediate testing.

Ask ChatGPT

You said:

for these alerts will i need to pay for them???? Alerts: SMTP (email) or Twilio / WhatsApp Cloud API for WhatsApp alerts

ChatGPT said:

Let's break it down clearly for you:

1 SMTP (Email Alerts)

- You can send emails for free using services like **Gmail SMTP** or **Outlook SMTP** — no payment required unless you send a *very high volume* of emails daily.
 - Example: Using Gmail, you can send **up to 500 emails/day** for free (if you enable *App Passwords* in your Google Account).
 - Tools: `smtpplib` in Python, PHPMailer in PHP.
 - **Cost:** \$0 for small-scale use.
-

2 Twilio (WhatsApp Alerts)

- **Twilio** offers a *free trial*, but after that, you pay per message (about **\$0.005–\$0.07** per message depending on location).
- You also need to verify your WhatsApp business number.
- **Best for:** Professional-scale messaging.

3 WhatsApp Cloud API (Meta)

- Meta's official WhatsApp Cloud API is **free for the first 1,000 conversations/month**.
- After that, you pay per conversation depending on region (usually **\$0.01–\$0.10** per chat session).
- **Best for:** Direct integration with WhatsApp, more scalable than Twilio.

✓ Recommendation for your simple healthcare system

- **Use SMTP for free email alerts** to patients/doctors.
- Add **WhatsApp Cloud API** later when you want to scale and reach patients more instantly.

If you want, I can give you a **small working Python script** for sending free alerts via Gmail SMTP so you can integrate it right away.

Do you want me to prepare that?