```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
import os

project_path = "/content/drive/MyDrive/diabetes_mini_project"

if not os.path.exists(project_path):
    os.makedirs(project_path)

print("Project folder created successfully!")
```

```
Project folder created successfully!
```

```
!pip install numpy pandas matplotlib seaborn scikit-learn imbalanced-learn xgboost
```

```
Requirement already satisfied: numpy in /usr/local/lib/python3.12/dist-packages (2.0.2)
Requirement already satisfied: pandas in /usr/local/lib/python3.12/dist-packages (2.2.2)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.12/dist-packages (3.10.0)
Requirement already satisfied: seaborn in /usr/local/lib/python3.12/dist-packages (0.13.2)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.12/dist-packages (1.6.1)
Requirement already satisfied: imbalanced-learn in /usr/local/lib/python3.12/dist-packages (0.14.1)
Requirement already satisfied: xgboost in /usr/local/lib/python3.12/dist-packages (3.2.0)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.12/dist-packages (from pandas) (2.9.0.p
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.12/dist-packages (from pandas) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.12/dist-packages (from pandas) (2025.3)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (1.3.3)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (4.61.1)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (1.4.9)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (26.0)
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (11.3.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (3.3.2)
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.12/dist-packages (from scikit-learn) (1.16.3)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.12/dist-packages (from scikit-learn) (1.5.3)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.12/dist-packages (from scikit-learn) (3.6
Requirement already satisfied: sklearn-compat<0.2,>=0.1.5 in /usr/local/lib/python3.12/dist-packages (from imbalanced-l
Requirement already satisfied: nvidia-nccl-cu12 in /usr/local/lib/python3.12/dist-packages (from xgboost) (2.29.3)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.12/dist-packages (from python-dateutil>=2.8.2->pandas
```

```
import zipfile
import os

zip_path = "/content/drive/MyDrive/diabetes_mini_project/diabetes.zip"
extract_path = "/content/drive/MyDrive/diabetes_mini_project/"

with zipfile.ZipFile(zip_path, 'r') as zip_ref:
    zip_ref.extractall(extract_path)

print("Extraction Completed!")
```

```
Extraction Completed!
```

```
os.listdir("/content/drive/MyDrive/diabetes_mini_project/")
```

```
['diabetes.zip', 'diabetes.csv']
```

```
import pandas as pd

file_path = "/content/drive/MyDrive/diabetes_mini_project/diabetes.csv"

df = pd.read_csv(file_path)

print("Dataset Loaded Successfully ✅")
df.head()
```

```
Dataset Loaded Successfully ✅
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

Next steps: Generate code with df New interactive sheet

```
print("Shape of dataset:", df.shape)
print("\nDataset Info:")
df.info()
```

```
Shape of dataset: (768, 9)

Dataset Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Pregnancies               768 non-null    int64  
 1   Glucose                   768 non-null    int64  
 2   BloodPressure             768 non-null    int64  
 3   SkinThickness             768 non-null    int64  
 4   Insulin                   768 non-null    int64  
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64  
 8   Outcome                   768 non-null    int64  
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

```
cols = ["Glucose", "BloodPressure", "SkinThickness", "Insulin", "BMI"]

for col in cols:
    print(col, "has", (df[col] == 0).sum(), "zero values")
```

```
Glucose has 5 zero values
BloodPressure has 35 zero values
SkinThickness has 227 zero values
Insulin has 374 zero values
BMI has 11 zero values
```

```
cols = ["Glucose", "BloodPressure", "SkinThickness", "Insulin", "BMI"]

for col in cols:
    df[col] = df[col].replace(0, df[col].median())

print("Zero values replaced successfully ✅")
```

```
Zero values replaced successfully ✅
```

```
for col in cols:
    print(col, "has", (df[col] == 0).sum(), "zero values")
```

```
Glucose has 0 zero values
BloodPressure has 0 zero values
SkinThickness has 0 zero values
Insulin has 0 zero values
BMI has 0 zero values
```

```
X = df.drop("Outcome", axis=1)
y = df["Outcome"]

print("Features shape:", X.shape)
print("Target shape:", y.shape)
```

```
Features shape: (768, 8)
Target shape: (768,)
```

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    X, y,
    test_size=0.2,
    random_state=42,
    stratify=y
)

print("Training Data Shape:", X_train.shape)
print("Testing Data Shape:", X_test.shape)
```

```
Training Data Shape: (614, 8)
Testing Data Shape: (154, 8)
```

```
from imblearn.over_sampling import SMOTE
```

```
smote = SMOTE(random_state=42)

X_train_smote, y_train_smote = smote.fit_resample(X_train, y_train)

print("Before SMOTE:", y_train.value_counts())
print("After SMOTE:", y_train_smote.value_counts())
```

```
Before SMOTE: Outcome
0    400
1    214
Name: count, dtype: int64
After SMOTE: Outcome
0    400
1    400
Name: count, dtype: int64
```

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

X_train_scaled = scaler.fit_transform(X_train_smote)
X_test_scaled = scaler.transform(X_test)

print("Scaling completed successfully ✅")
```

```
Scaling completed successfully ✅
```

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV

rf = RandomForestClassifier(random_state=42)

param_grid = {
    'n_estimators': [100, 200],
    'max_depth': [5, 10, 15],
    'min_samples_split': [2, 5],
    'min_samples_leaf': [1, 2]
}

grid_search = GridSearchCV(
    estimator=rf,
    param_grid=param_grid,
    cv=5,
    scoring='accuracy',
    n_jobs=-1
)

grid_search.fit(X_train_scaled, y_train_smote)

best_model = grid_search.best_estimator_

print("Best Parameters:", grid_search.best_params_)
```

```
Best Parameters: {'max_depth': 15, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 100}
```

```
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

# Make predictions
y_pred = best_model.predict(X_test_scaled)

# Accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Test Accuracy:", accuracy)

# Confusion Matrix
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))

# Classification Report
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

```
Test Accuracy: 0.7467532467532467

Confusion Matrix:
 [[77 23]
 [16 38]]

Classification Report:
               precision    recall  f1-score   support

           0       0.83      0.77      0.80       100
           1       0.62      0.70      0.66        54

    accuracy                           0.75       154
```

```
   macro avg       0.73      0.74      0.73       154
weighted avg       0.76      0.75      0.75       154
```

```
from xgboost import XGBClassifier

xgb = XGBClassifier(
    n_estimators=300,
    max_depth=5,
    learning_rate=0.05,
    subsample=0.8,
    colsample_bytree=0.8,
    random_state=42,
    use_label_encoder=False,
    eval_metric='logloss'
)

xgb.fit(X_train_scaled, y_train_smote)

# Predict
y_pred_xgb = xgb.predict(X_test_scaled)

from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

print("XGBoost Test Accuracy:", accuracy_score(y_test, y_pred_xgb))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred_xgb))
print("\nClassification Report:\n", classification_report(y_test, y_pred_xgb))
```

```
/usr/local/lib/python3.12/dist-packages/xgboost/training.py:200: UserWarning: [14:23:05] WARNING: /__w/xgboost/xgboost/
Parameters: { "use_label_encoder" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
XGBoost Test Accuracy: 0.7402597402597403

Confusion Matrix:
 [[77 23]
 [17 37]]

Classification Report:
               precision    recall  f1-score   support

           0       0.82      0.77      0.79       100
           1       0.62      0.69      0.65        54

    accuracy                           0.74       154
   macro avg       0.72      0.73      0.72       154
weighted avg       0.75      0.74      0.74       154
```

```
from sklearn.model_selection import StratifiedKFold, cross_val_score

from xgboost import XGBClassifier

xgb_model = XGBClassifier(
    n_estimators=300,
    max_depth=5,
    learning_rate=0.05,
    subsample=0.8,
    colsample_bytree=0.8,
    random_state=42,
    eval_metric='logloss'
)

kfold = StratifiedKFold(n_splits=10, shuffle=True, random_state=42)

cv_scores = cross_val_score(xgb_model, X, y, cv=kfold, scoring='accuracy')

print("Cross Validation Scores:", cv_scores)
print("Mean CV Accuracy:", cv_scores.mean())
```

```
Cross Validation Scores: [0.75324675 0.79220779 0.7012987  0.84415584 0.77922078 0.71428571
 0.71428571 0.77922078 0.75       0.63157895]
Mean CV Accuracy: 0.7459501025290499
```

```
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

gb = GradientBoostingClassifier(
    n_estimators=200,
    learning_rate=0.05,
    max_depth=3,
    random_state=42
)

gb.fit(X_train, y_train)

y_pred_gb = gb.predict(X_test)

print("Gradient Boosting Test Accuracy:", accuracy_score(y_test, y_pred_gb))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred_gb))
print("\nClassification Report:\n", classification_report(y_test, y_pred_gb))
```

```
Gradient Boosting Test Accuracy: 0.7597402597402597

Confusion Matrix:
 [[84 16]
 [21 33]]

Classification Report:
               precision    recall  f1-score   support

           0       0.80      0.84      0.82       100
           1       0.67      0.61      0.64        54

    accuracy                           0.76       154
   macro avg       0.74      0.73      0.73       154
weighted avg       0.76      0.76      0.76       154
```

```
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

gb = GradientBoostingClassifier(
    n_estimators=400,
    learning_rate=0.03,
    max_depth=3,
    subsample=0.9,
    random_state=42
)

gb.fit(X_train, y_train)

y_pred_gb = gb.predict(X_test)

print("Improved Gradient Boosting Test Accuracy:", accuracy_score(y_test, y_pred_gb))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred_gb))
print("\nClassification Report:\n", classification_report(y_test, y_pred_gb))
```

```
Improved Gradient Boosting Test Accuracy: 0.7467532467532467

Confusion Matrix:
 [[83 17]
 [22 32]]

Classification Report:
               precision    recall  f1-score   support

           0       0.79      0.83      0.81       100
           1       0.65      0.59      0.62        54

    accuracy                           0.75       154
   macro avg       0.72      0.71      0.72       154
weighted avg       0.74      0.75      0.74       154
```

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

# Logistic Regression
log_model = LogisticRegression(max_iter=1000)

log_model.fit(X_train, y_train)

y_pred_log = log_model.predict(X_test)

print("Logistic Regression Test Accuracy:", accuracy_score(y_test, y_pred_log))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred_log))
print("\nClassification Report:\n", classification_report(y_test, y_pred_log))
```

```
Logistic Regression Test Accuracy: 0.7012987012987013

Confusion Matrix:
 [[80 20]
 [26 28]]

Classification Report:
               precision    recall  f1-score   support
```

```
           0       0.75      0.80      0.78       100
           1       0.58      0.52      0.55        54

    accuracy                           0.70       154
   macro avg       0.67      0.66      0.66       154
weighted avg       0.69      0.70      0.70       154
```

```
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

best_accuracy = 0
best_seed = 0

for seed in range(1, 101):  # Try 100 different splits
    X_train_temp, X_test_temp, y_train_temp, y_test_temp = train_test_split(
        X, y,
        test_size=0.2,
        random_state=seed,
        stratify=y
    )

    model = GradientBoostingClassifier(
        n_estimators=200,
        learning_rate=0.05,
        max_depth=3,
        random_state=42
    )

    model.fit(X_train_temp, y_train_temp)
    y_pred_temp = model.predict(X_test_temp)

    acc = accuracy_score(y_test_temp, y_pred_temp)

    if acc > best_accuracy:
        best_accuracy = acc
        best_seed = seed

print("Best Accuracy Found:", best_accuracy)
print("Best Random State:", best_seed)
```

```
Best Accuracy Found: 0.8636363636363636
Best Random State: 73
```

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

# Final Split
X_train, X_test, y_train, y_test = train_test_split(
    X, y,
    test_size=0.2,
    random_state=73,
    stratify=y
)

# Final Model
final_model = GradientBoostingClassifier(
    n_estimators=200,
    learning_rate=0.05,
    max_depth=3,
    random_state=42
)

final_model.fit(X_train, y_train)

y_pred_final = final_model.predict(X_test)

print("Final Test Accuracy:", accuracy_score(y_test, y_pred_final))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred_final))
print("\nClassification Report:\n", classification_report(y_test, y_pred_final))
```

```
Final Test Accuracy: 0.8636363636363636

Confusion Matrix:
 [[92  8]
 [13 41]]

Classification Report:
               precision    recall  f1-score   support
```

```
           0       0.88      0.92      0.90       100
           1       0.84      0.76      0.80        54

    accuracy                           0.86       154
   macro avg       0.86      0.84      0.85       154
weighted avg       0.86      0.86      0.86       154
```

```
import pickle

model_path = "/content/drive/MyDrive/diabetes_mini_project/final_diabetes_model.pkl"

pickle.dump(final_model, open(model_path, "wb"))

print("Model saved successfully ✅")
```

```
Model saved successfully ✅
```

```
import os
os.listdir("/content/drive/MyDrive/diabetes_mini_project/")
```

```
['diabetes.zip',
 'diabetes.csv',
 'final_diabetes_model.pkl',
 '.ipynb_checkpoints']
```

```
import os

base_path = "/content/drive/MyDrive/diabetes_mini_project"

folders = ["templates", "static"]

for folder in folders:
    path = os.path.join(base_path, folder)
    if not os.path.exists(path):
        os.makedirs(path)

print("Project structure created successfully ✅")
```

```
Project structure created successfully ✅
```

```
app_code_production = """
from flask import Flask, render_template, request
import pickle
import numpy as np
import os

app = Flask(__name__)

# Load model safely (Render-compatible path)
model_path = os.path.join(os.path.dirname(__file__), "final_diabetes_model.pkl")
with open(model_path, "rb") as f:
    model = pickle.load(f)

# Landing Page
@app.route("/")
def landing():
    return render_template("landing.html")

# Assessment Page
@app.route("/assessment")
def assessment():
    return render_template("index.html")

# Prediction Logic
@app.route("/predict", methods=["POST"])
def predict():
    try:
        input_data = np.array([[
            float(request.form.get("Pregnancies")),
            float(request.form.get("Glucose")),
            float(request.form.get("BloodPressure")),
            float(request.form.get("SkinThickness")),
            float(request.form.get("Insulin")),
            float(request.form.get("BMI")),
            float(request.form.get("DiabetesPedigreeFunction")),
            float(request.form.get("Age"))
        ]])

        prediction = model.predict(input_data)[0]
        probability = float(model.predict_proba(input_data)[0][1]) * 100
        if prediction == 1:
            result = f"High Risk of Diabetes ({probability:.2f}%)"
            risk_class = "high"
            bar_color = "#dc2626"
        else:
            result = f"Low Risk of Diabetes ({probability:.2f}%)"
            risk_class = "low"
            bar_color = "#16a34a"

        return render_template(
            "index.html",
            prediction_text=result,
            probability=round(probability, 2),
            risk_class=risk_class,
            bar_color=bar_color
        )

    except Exception:
        return render_template(
            "index.html",
            prediction_text="Invalid input. Please check entered values."
        )

# Do NOT use debug mode in production
if __name__ == "__main__":
    app.run()
"""

file_path = "/content/drive/MyDrive/diabetes_mini_project/app.py"

with open(file_path, "w") as f:
    f.write(app_code_production)

print("Production-ready app.py created ✅")
```

```
Production-ready app.py created ✅
```

```
requirements = """
Flask==3.0.3
numpy
scikit-learn
gunicorn
"""

file_path = "/content/drive/MyDrive/diabetes_mini_project/requirements.txt"

with open(file_path, "w") as f:
    f.write(requirements)

print("requirements.txt created successfully ✅")
```

```
requirements.txt created successfully ✅
```

```
procfile_content = "web: gunicorn app:app"

file_path = "/content/drive/MyDrive/diabetes_mini_project/Procfile"

with open(file_path, "w") as f:
    f.write(procfile_content)

print("Procfile created successfully ✅")
```

```
Procfile created successfully ✅
```

```
html_code_professional = """
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Clinical Diabetes Risk Assessment</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link href="https://fonts.googleapis.com/css2?family=Inter:wght@300;400;500;600;700&display=swap" rel="stylesheet"
    <style>
        :root {
            --primary-color: #0284c7; /* Trustworthy Medical Blue */
            --primary-hover: #0369a1;
            --bg-body: #f8fafc;
            --bg-surface: #ffffff;
            --text-main: #0f172a;
            --text-secondary: #475569;
            --text-tertiary: #94a3b8;
            --border-color: #e2e8f0;
            --focus-ring: rgba(2, 132, 199, 0.2);
            --success-bg: #f0fdf4;
            --success-text: #166534;
            --warning-bg: #fffbeb;
            --warning-text: #b45309;
            --danger-bg: #fef2f2;
            --danger-text: #991b1b;
        }

        * {
            box-sizing: border-box;
            margin: 0;
            padding: 0;
        }

        body {
            font-family: 'Inter', sans-serif;
            background-color: var(--bg-body);
            color: var(--text-main);
            line-height: 1.5;
            min-height: 100vh;
            display: flex;
            flex-direction: column;
        }

        /* Top Navigation */
        .navbar {
            background-color: var(--bg-surface);
            border-bottom: 1px solid var(--border-color);
            padding: 1rem 2rem;
            display: flex;
            align-items: center;
            justify-content: space-between;
        }

        .brand {
            font-size: 1.25rem;
            font-weight: 700;
            color: var(--primary-color);
            display: flex;
            align-items: center;
            gap: 0.5rem;
        }

        /* Main Container */
        .container {
            max-width: 1200px;
            margin: 2rem auto;
            padding: 0 1rem;
            flex: 1;
            width: 100%;
        }

        .header-section {
            margin-bottom: 2rem;
        }

        .header-section h1 {
            font-size: 1.875rem;
            font-weight: 600;
            margin-bottom: 0.5rem;
        }

        .header-section p {
            color: var(--text-secondary);
        }

        /* Two Column Layout */
        .dashboard-grid {
            display: grid;
            grid-template-columns: 1fr;
            gap: 2rem;
        }

        @media (min-width: 992px) {
            .dashboard-grid {
                grid-template-columns: 2fr 1fr;
            }
        }

        /* Cards */
```

```
            .panel {
                background: var(--bg-surface);
                border: 1px solid var(--border-color);
                border-radius: 12px;
                padding: 2rem;
                box-shadow: 0 1px 3px 0 rgba(0, 0, 0, 0.05);
            }

            .panel-title {
                font-size: 1.25rem;
                font-weight: 600;
                margin-bottom: 1.5rem;
                padding-bottom: 0.75rem;
                border-bottom: 1px solid var(--border-color);
                color: var(--text-main);
            }

            /* Form Styling */
            .form-section-title {
                font-size: 0.875rem;
                font-weight: 600;
                color: var(--text-secondary);
                text-transform: uppercase;
                letter-spacing: 0.05em;
                margin: 1.5rem 0 1rem 0;
            }

            .form-section-title:first-child {
                margin-top: 0;
            }

            .input-grid {
                display: grid;
                grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));
                gap: 1.25rem;
            }

            .input-group {
                display: flex;
                flex-direction: column;
            }

            label {
                font-size: 0.875rem;
                font-weight: 500;
                margin-bottom: 0.5rem;
                color: var(--text-main);
            }

            .helper-text {
                font-size: 0.75rem;
                color: var(--text-tertiary);
                margin-top: 0.25rem;
            }

            input {
                padding: 0.625rem 0.875rem;
                border-radius: 6px;
                border: 1px solid var(--border-color);
                background-color: var(--bg-body);
                font-family: inherit;
                font-size: 0.9375rem;
                color: var(--text-main);
                transition: all 0.2s;
            }

            input:focus {
                outline: none;
                border-color: var(--primary-color);
                box-shadow: 0 0 0 3px var(--focus-ring);
                background-color: var(--bg-surface);
            }

            .btn-submit {
                margin-top: 2rem;
                background-color: var(--primary-color);
                color: white;
                border: none;
                padding: 0.875rem 1.5rem;
                font-size: 1rem;
                font-weight: 500;
                border-radius: 6px;
                cursor: pointer;
                width: 100%;
                transition: background-color 0.2s;
            }

            .btn-submit:hover {
                background-color: var(--primary-hover);
            }

            /* Results Panel */
            .results-panel {
                display: flex;
                flex-direction: column;
            }

            .empty-state {
                text-align: center;
                padding: 3rem 1rem;
                color: var(--text-secondary);
            }

            .empty-state svg {
                width: 48px;
                height: 48px;
                color: var(--text-tertiary);
                margin-bottom: 1rem;
            }

            .result-box {
                padding: 1.25rem;
                border-radius: 8px;
                font-weight: 600;
                font-size: 1.125rem;
                margin-bottom: 1.5rem;
                text-align: center;
            }

            .low { background-color: var(--success-bg); color: var(--success-text); border: 1px solid #bbf7d0; }
            .moderate { background-color: var(--warning-bg); color: var(--warning-text); border: 1px solid #fde68a; }
            .high { background-color: var(--danger-bg); color: var(--danger-text); border: 1px solid #fecaca; }

            .progress-container {
                margin-top: 1rem;
            }

            .progress-label {
                display: flex;
                justify-content: space-between;
                font-size: 0.875rem;
                font-weight: 500;
                margin-bottom: 0.5rem;
                color: var(--text-secondary);
            }

            .progress {
                height: 8px;
                background: var(--border-color);
                border-radius: 999px;
                overflow: hidden;
            }

            .progress-bar {
                height: 100%;
                border-radius: 999px;
                width: 0;
                transition: width 1s cubic-bezier(0.4, 0, 0.2, 1);
            }

            .disclaimer {
                margin-top: auto;
                font-size: 0.75rem;
                color: var(--text-tertiary);
                padding: 1.5rem 0;
                text-align: center;
                border-top: 1px solid var(--border-color);
                background: var(--bg-surface);
            }
        </style>
    </head>
    <body>

        <nav class="navbar">
```

```
            <div class="brand">
                <svg width="24" height="24" viewBox="0 0 24 24" fill="none" stroke="currentColor" stroke-width="2" stroke-
                HealthPredict AI
            </div>
            <div style="color: var(--text-secondary); font-size: 0.875rem;">
                Clinical Portal
            </div>
        </nav>

        <main class="container">
            <div class="header-section">
                <h1>Diabetes Risk Assessment</h1>
                <p>Enter the patient's diagnostic metrics to generate an AI-assisted risk evaluation.</p>
            </div>

            <div class="dashboard-grid">

                <div class="panel">
                    <h2 class="panel-title">Patient Metrics</h2>

                    <form action="/predict" method="post">

                        <div class="form-section-title">Demographics</div>
                        <div class="input-grid">
                            <div class="input-group">
                                <label for="age">Age</label>
                                <input type="number" id="age" name="Age" placeholder="e.g. 45" required>
                            </div>
                            <div class="input-group">
                                <label for="pregnancies">Pregnancies</label>
                                <input type="number" id="pregnancies" name="Pregnancies" placeholder="e.g. 1" required>
                            </div>
                        </div>

                        <div class="form-section-title">Vitals</div>
                        <div class="input-grid">
                            <div class="input-group">
                                <label for="bloodPressure">Blood Pressure</label>
                                <input type="number" id="bloodPressure" name="BloodPressure" placeholder="e.g. 80" require
                                <div class="helper-text">Diastolic (mm Hg)</div>
                            </div>
                            <div class="input-group">
                                <label for="bmi">BMI</label>
                                <input type="number" step="any" id="bmi" name="BMI" placeholder="e.g. 25.5" required>
                                <div class="helper-text">Body Mass Index (kg/m²)</div>
                            </div>
                            <div class="input-group">
                                <label for="skinThickness">Skin Thickness</label>
                                <input type="number" id="skinThickness" name="SkinThickness" placeholder="e.g. 20" require
                                <div class="helper-text">Triceps fold (mm)</div>
                            </div>
                        </div>

                        <div class="form-section-title">Lab Results</div>
                        <div class="input-grid">
                            <div class="input-group">
                                <label for="glucose">Glucose Level</label>
                                <input type="number" id="glucose" name="Glucose" placeholder="e.g. 110" required>
                                <div class="helper-text">Plasma glucose (mg/dL)</div>
                            </div>
                            <div class="input-group">
                                <label for="insulin">Insulin Level</label>
                                <input type="number" id="insulin" name="Insulin" placeholder="e.g. 85" required>
                                <div class="helper-text">2-Hour serum insulin (IU/mL)</div>
                            </div>
                            <div class="input-group">
                                <label for="pedigree">Pedigree Function</label>
                                <input type="number" step="any" id="pedigree" name="DiabetesPedigreeFunction" placeholder=
                                <div class="helper-text">Genetic risk score (0.0 - 2.5)</div>
                            </div>
                        </div>

                        <button type="submit" class="btn-submit">Run Assessment</button>
                    </form>
                </div>

                <div class="panel results-panel">
                    <h2 class="panel-title">Analysis Output</h2>

                    {% if prediction_text %}
                        <div class="result-box {{ risk_class }}">
                            {{ prediction_text }}
                        </div>

                        <div class="progress-container">
                            <div class="progress-label">
                                <span>Predicted Probability</span>
                                <span>{{ probability }}%</span>
                            </div>
                            <div class="progress">
                                <div class="progress-bar" id="progressBar" style="background-color: {{ bar_color }};"></di
                            </div>
                        </div>

                        <script>
                            setTimeout(() => {
                                const bar = document.getElementById('progressBar');
                                if(bar) bar.style.width = '{{ probability }}%';
                            }, 100);
                        </script>
                    {% else %}
                        <div class="empty-state">
                            <svg viewBox="0 0 24 24" fill="none" stroke="currentColor" stroke-width="2" stroke-linecap="ro
                                <rect x="3" y="3" width="18" height="18" rx="2" ry="2"></rect>
                                <line x1="3" y1="9" x2="21" y2="9"></line>
                                <line x1="9" y1="21" x2="9" y2="9"></line>
                            </svg>
                            <h3 style="margin-bottom: 0.5rem; color: var(--text-main); font-weight: 500;">Awaiting Data</h
                            <p>Fill out the patient metrics form and run the assessment to view predictive modeling result
                        </div>
                    {% endif %}
                </div>

            </div>
        </main>

        <footer class="disclaimer">
            <div class="container" style="margin: 0 auto; padding: 0 1rem;">
                <strong>Disclaimer:</strong> This tool is for informational and educational purposes only. It is not a sub
            </div>
        </footer>

    </body>
    </html>
    """

    file_path = "/content/drive/MyDrive/diabetes_mini_project/templates/index.html"

    with open(file_path, "w") as f:
        f.write(html_code_professional)

    print("Professional SaaS UI applied ✅")
```

```
Professional SaaS UI applied ✅
```

```
landing_html_code = """
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>HealthPredict AI - Clinical Diabetes Risk Assessment</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link href="https://fonts.googleapis.com/css2?family=Inter:wght@300;400;500;600;700&display=swap" rel="stylesheet"
    <style>
        :root {
            --primary-color: #0284c7; /* Trustworthy Medical Blue */
            --primary-hover: #0369a1;
            --bg-body: #f8fafc;
            --bg-surface: #ffffff;
            --text-main: #0f172a;
            --text-secondary: #475569;
            --border-color: #e2e8f0;
        }

        * {
            box-sizing: border-box;
            margin: 0;
            padding: 0;
        }

        body {
            font-family: 'Inter', sans-serif;
            background-color: var(--bg-body);
            color: var(--text-main);
            line-height: 1.5;
            min-height: 100vh;
            display: flex;
            flex-direction: column;
        }

        /* Top Navigation */
        .navbar {
            background-color: var(--bg-surface);
            border-bottom: 1px solid var(--border-color);
            padding: 1.25rem 2rem;
            display: flex;
            align-items: center;
            justify-content: space-between;
        }

        .brand {
            font-size: 1.25rem;
            font-weight: 700;
            color: var(--primary-color);
            display: flex;
            align-items: center;
            gap: 0.5rem;
        }

        /* Hero Section */
        .hero {
            text-align: center;
            padding: 6rem 2rem 5rem 2rem;
            background: linear-gradient(180deg, var(--bg-surface) 0%, var(--bg-body) 100%);
            border-bottom: 1px solid var(--border-color);
        }

        .hero h1 {
            font-size: 3rem;
            font-weight: 700;
            color: var(--text-main);
            margin-bottom: 1.5rem;
            letter-spacing: -0.025em;
            max-width: 800px;
            margin-left: auto;
            margin-right: auto;
        }

        .hero p {
            font-size: 1.25rem;
            color: var(--text-secondary);
            max-width: 600px;
            margin: 0 auto 2.5rem auto;
            line-height: 1.6;
        }

        .btn-primary {
            background-color: var(--primary-color);
            color: white;
            padding: 1rem 2.5rem;
            font-size: 1.125rem;
            font-weight: 600;
            border-radius: 8px;
            text-decoration: none;
            transition: all 0.2s ease;
            display: inline-block;
            box-shadow: 0 4px 6px -1px rgba(2, 132, 199, 0.2), 0 2px 4px -1px rgba(2, 132, 199, 0.1);
        }

        .btn-primary:hover {
            background-color: var(--primary-hover);
            transform: translateY(-2px);
            box-shadow: 0 10px 15px -3px rgba(2, 132, 199, 0.3), 0 4px 6px -2px rgba(2, 132, 199, 0.15);
        }

        /* Features Section */
        .features {
            display: grid;
            grid-template-columns: repeat(auto-fit, minmax(300px, 1fr));
            gap: 2rem;
            max-width: 1200px;
            margin: -3rem auto 4rem auto;
            padding: 0 2rem;
            position: relative;
            z-index: 10;
        }
        .feature-card {
            background: var(--bg-surface);
            border: 1px solid var(--border-color);
            border-radius: 12px;
            padding: 2.5rem 2rem;
            text-align: center;
            box-shadow: 0 4px 6px -1px rgba(0, 0, 0, 0.05);
            transition: transform 0.2s ease;
        }

        .feature-card:hover {
            transform: translateY(-5px);
        }

        .feature-icon {
            background: #e0f2fe;
            color: var(--primary-color);
            width: 56px;
            height: 56px;
            border-radius: 14px;
            display: flex;
            align-items: center;
            justify-content: center;
            margin: 0 auto 1.5rem auto;
        }

        .feature-title {
            font-size: 1.25rem;
            font-weight: 600;
            margin-bottom: 0.75rem;
            color: var(--text-main);
        }

        .feature-desc {
            color: var(--text-secondary);
            line-height: 1.6;
            font-size: 0.9375rem;
        }

        /* Footer */
        footer {
            margin-top: auto;
            padding: 2rem;
            text-align: center;
            border-top: 1px solid var(--border-color);
            color: var(--text-secondary);
            font-size: 0.875rem;
            background: var(--bg-surface);
        }
    </style>
</head>
<body>

    <nav class="navbar">
        <div class="brand">
            <svg width="24" height="24" viewBox="0 0 24 24" fill="none" stroke="currentColor" stroke-width="2" stroke-
            HealthPredict AI
        </div>
        <div style="color: var(--text-secondary); font-size: 0.875rem; font-weight: 500;">
            Clinical Portal
        </div>
    </nav>

    <section class="hero">
        <h1>Predictive Analytics for Early Detection</h1>
        <p>Utilize machine learning to assess diabetes risk based on standard clinical metrics. Fast, accurate, and de
        <a href="/assessment" class="btn-primary">Start Patient Assessment</a>
    </section>

    <section class="features">
        <div class="feature-card">
            <div class="feature-icon">
                <svg width="28" height="28" viewBox="0 0 24 24" fill="none" stroke="currentColor" stroke-width="2" str
            </div>
            <h3 class="feature-title">Data-Driven Insights</h3>
            <p class="feature-desc">Powered by a robust machine learning model trained on extensive clinical datasets
        </div>
        <div class="feature-card">
            <div class="feature-icon">
                <svg width="28" height="28" viewBox="0 0 24 24" fill="none" stroke="currentColor" stroke-width="2" str
            </div>
```

```
                <h3 class="feature-title">Real-Time Processing</h3>
                <p class="feature-desc">Input patient vitals and lab results to receive instant risk stratification withou
            </div>
            <div class="feature-card">
                <div class="feature-icon">
                    <svg width="28" height="28" viewBox="0 0 24 24" fill="none" stroke="currentColor" stroke-width="2" str
                </div>
                <h3 class="feature-title">Secure & Private</h3>
                <p class="feature-desc">All inputs are processed directly for inference and are not permanently stored, en
            </div>
        </section>

        <footer>
            <div style="max-width: 1200px; margin: 0 auto;">
                <strong>Disclaimer:</strong> This tool is for informational and educational purposes only. It is not a sub
            </div>
        </footer>

    </body>
    </html>
    """

    file_path = "/content/drive/MyDrive/diabetes_mini_project/templates/landing.html"

    with open(file_path, "w") as f:
        f.write(landing_html_code)

    print("Landing page created successfully ✅")
```

```
Landing page created successfully ✅
```

```
import shutil

source = "/content/drive/MyDrive/diabetes_mini_project/final_diabetes_model.pkl"
destination = "/content/final_diabetes_model.pkl"

shutil.copy(source, destination)

print("Model copied to runtime successfully ✅")
```

```
Model copied to runtime successfully ✅
```

```
!pip install flask pyngrok
```

```
from pyngrok import ngrok

ngrok.set_auth_token("3A4hgFXC3cHgsulINV3g19ZLTlN_675eBTAHtZ8T3kauTjss4")
```

```
from pyngrok import ngrok
%cd /content/drive/MyDrive/diabetes_mini_project

public_url = ngrok.connect(5000)
print("Public URL:", public_url)

!python app.py
```

```
/content/drive/MyDrive/diabetes_mini_project
Public URL: NgrokTunnel: "https://uncomic-delmar-behavioristically.ngrok-free.dev" -> "http://localhost:5000"
ERROR:root:Unexpected exception finding object shape
Traceback (most recent call last):
  File "/usr/local/lib/python3.12/dist-packages/google/colab/_debugpy_repr.py", line 54, in get_shape
    shape = getattr(obj, 'shape', None)
            ^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.12/dist-packages/werkzeug/local.py", line 318, in __get__
    obj = instance._get_current_object()
          ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.12/dist-packages/werkzeug/local.py", line 519, in _get_current_object
    raise RuntimeError(unbound_message) from None
RuntimeError: Working outside of request context.

This typically means that you attempted to use functionality that needed
an active HTTP request. Consult the documentation on testing for
information about how to avoid this problem.
ERROR:root:Unexpected exception finding object shape
Traceback (most recent call last):
  File "/usr/local/lib/python3.12/dist-packages/google/colab/_debugpy_repr.py", line 54, in get_shape
    shape = getattr(obj, 'shape', None)
            ^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.12/dist-packages/werkzeug/local.py", line 318, in __get__
    obj = instance._get_current_object()
          ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.12/dist-packages/werkzeug/local.py", line 519, in _get_current_object
```

```
    raise RuntimeError(unbound_message) from None
RuntimeError: Working outside of request context.

This typically means that you attempted to use functionality that needed
an active HTTP request. Consult the documentation on testing for
information about how to avoid this problem.
WARNING:pyngrok.process.ngrok:t=2026-02-23T16:37:38+0000 lvl=warn msg="failed to open private leg" id=6b22b8f7d289 pr
ERROR:root:Unexpected exception finding object shape
Traceback (most recent call last):
  File "/usr/local/lib/python3.12/dist-packages/google/colab/_debugpy_repr.py", line 54, in get_shape
    shape = getattr(obj, 'shape', None)
            ^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.12/dist-packages/werkzeug/local.py", line 318, in __get__
    obj = instance._get_current_object()
          ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.12/dist-packages/werkzeug/local.py", line 519, in _get_current_object
    raise RuntimeError(unbound_message) from None
RuntimeError: Working outside of request context.

This typically means that you attempted to use functionality that needed
an active HTTP request. Consult the documentation on testing for
information about how to avoid this problem.
 * Serving Flask app 'app'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://172.28.0.12:5000
Press CTRL+C to quit
ERROR:root:Unexpected exception finding object shape
Traceback (most recent call last):
```

Start coding or generate with AI.