In [7]:

```
from pathlib import Path
import sys, subprocess, warnings

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

def pip_import(pkg):
    try:
        return __import__(pkg)
    except ModuleNotFoundError:
        print(f"[INFO] Installing {pkg} …")
        subprocess.check_call([sys.executable, "-m", "pip", "install",
        return __import__(pkg)

wordcloud = pip_import("wordcloud")
shap      = pip_import("shap")
sklearn   = pip_import("sklearn")

from wordcloud import WordCloud, STOPWORDS
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import make_pipeline
from sklearn.model_selection import train_test_split

warnings.filterwarnings("ignore")
plt.style.use("default")
```

In [8]:

```
# 1 Load data
CSV = Path("fake_job_postings.csv")
if not CSV.is_file():
    sys.exit(f"[ERROR] {CSV.name} not found in {Path.cwd()}")

df = pd.read_csv(CSV)
print(f"Loaded {len(df):,} rows  |  {len(df.columns)} columns")
```

```
Loaded 17,880 rows  |  18 columns
```

In [9]:

```
# 2 Basic cleaning
if df.columns.duplicated().any():
    df = df.loc[:, ~df.columns.duplicated()].copy()

LABEL = "fraudulent"
df[LABEL] = df[LABEL].astype(int)

before = len(df)
df = df.drop_duplicates()
print("Duplicates removed:", before - len(df))
```

```
Duplicates removed: 0
```

In [10]:

```
# 3 Feature engineering
TEXT_FIELDS = ["title", "company_profile", "description", "requiremen
for col in TEXT_FIELDS:
    df[f"len_{col}"] = df[col].fillna("").str.len()

BINARY_COLS = [c for c in ["telecommuting", "has_company_logo", "has_
NUM_FEATURES = [f"len_{c}" for c in TEXT_FIELDS] + BINARY_COLS
```

In [11]:

```
# 4 Visuals
# Figure 1: class balance
class_counts = df[LABEL].value_counts().rename({0:"real", 1:"fake"})
class_counts.plot(kind="bar", color=["steelblue", "indianred"], figsi
plt.title("Real vs. Fake Job Postings"); plt.ylabel("Count"); plt.xti
plt.tight_layout(); plt.savefig("class_balance.png"); plt.close()

# Figure 2: text-length distributions
plt.figure(figsize=(8,6))
for field, color in zip(["description", "requirements"], ["steelblue",
    sns.kdeplot(df[f"len_{field}"], fill=True, alpha=0.3, label=field,
plt.xlim(0,5000); plt.xlabel("Character Count"); plt.title("Text-Leng
plt.legend(); plt.tight_layout(); plt.savefig("text_length_dist.png");

# Figure 3: correlation heat-map
plt.figure(figsize=(8,6))
corr = df[[LABEL] + NUM_FEATURES].corr().round(2)
sns.heatmap(corr, annot=True, cmap="coolwarm", vmin=-1, vmax=1)
plt.title("Correlation Matrix vs. Fraudulent Flag")
plt.tight_layout(); plt.savefig("correlation_heatmap.png"); plt.close

# Figure 4: word clouds
for cls, fname in [(0,"top_words_real.png"),(1,"top_words_fake.png")]:
    blob = " ".join(df.loc[df[LABEL]==cls, "description"].fillna("").
    WordCloud(width=800, height=400, background_color="white", stopwo
```

In [16]:

```
# 0. Create one combined text column:
df["text_all"] = (
    df["title"].fillna("") + " " +
    df["description"].fillna("") + " " +
    df["requirements"].fillna("")
)

# 5 Quick model – now using a single series, not a DataFrame:
df["text_all"] = df["text_all"].fillna("")    # ensure no nulls
X_train, X_test, y_train, y_test = train_test_split(
    df["text_all"],               # <-- single pandas Series
    df[LABEL],                    # your fraud label
    test_size=0.2,
    stratify=df[LABEL],
    random_state=42
)

model = make_pipeline(
    TfidfVectorizer(
        max_features=10000,
        ngram_range=(1,2),
        stop_words="english"
    ),
    LogisticRegression(
        max_iter=1000,
        class_weight="balanced"
    )
)
model.fit(X_train, y_train)
print("Test accuracy:", round(model.score(X_test, y_test), 3))

# 5a. Generate predictions & probabilities
y_pred  = model.predict(X_test)
y_proba = model.predict_proba(X_test)[:,1]



# 5b. Confusion Matrix
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
cm = confusion_matrix(y_test, y_pred, labels=[0,1])
disp = ConfusionMatrixDisplay(cm, display_labels=["Real","Fake"])
disp.plot(cmap="Blues")
plt.title("Confusion Matrix on Hold-Out Set")
plt.tight_layout()
plt.savefig("confusion_matrix.png")
plt.close()


# 5c. ROC Curve
from sklearn.metrics import roc_curve, roc_auc_score, RocCurveDisplay
fpr, tpr, _ = roc_curve(y_test, y_proba)
auc = roc_auc_score(y_test, y_proba)
RocCurveDisplay(fpr=fpr, tpr=tpr, roc_auc=auc).plot()
plt.title(f"ROC Curve (AUC = {auc:.2f})")
plt.tight_layout()
plt.savefig("roc_curve.png")
plt.close()
```

```
Test accuracy: 0.962
```

In [17]:

```
# 6 SHAP explanation for a representative fake ad
explainer = shap.LinearExplainer(
    model.named_steps["logisticregression"],
    model.named_steps["tfidfvectorizer"].transform(X_train),
    feature_perturbation="interventional"
)
idx    = X_test[y_test==1].index[0]
row    = X_test.loc[idx]
vec    = model.named_steps["tfidfvectorizer"].transform([row])
shap_values = explainer(vec)
plt.figure(figsize=(10,4))
shap.plots.waterfall(shap_values[0], max_display=12, show=False)
plt.title("Why this listing is predicted fake")
plt.tight_layout()
plt.savefig("shap_waterfall.png")
plt.close()
```

In [ ]: