```
# app.py


import yfinance as yf
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from keras.models import Sequential
from keras.layers import LSTM, Dense, Input
from keras.models import load_model
import streamlit as st
import hashlib
import json
import os
import datetime
import requests


# ------------------------- User Auth -------------------------
USER_FILE = "users.json"


USER_PROFILES = {
    "Apple": {"name": "Apple Inc.", "age": "47", "number": "+1-800-275-2273"},
    "Tesla": {"name": "Tesla Inc.", "age": "21", "number": "+1-650-681-5000"},
    "Infosys": {"name": "Infosys Ltd.", "age": "43", "number": "+91-80-2852-0261"},
    "Tata": {"name": "Tata Consultancy Services", "age": "56", "number": "+91-22-6778-9999"},
}

def load_users():
    if os.path.exists(USER_FILE):
        with open(USER_FILE, "r") as f:
            return json.load(f)
    return {}

def save_users(users):
    with open(USER_FILE, "w") as f:
        json.dump(users, f)

def hash_password(password):
    return hashlib.sha256(password.encode()).hexdigest()

def signup_user(username, password):
    users = load_users()
    if username in users:
        return False
    users[username] = hash_password(password)
    save_users(users)
    return True

def check_login(username, password):
    users = load_users()
    return username in users and users[username] == hash_password(password)


def change_password(username, new_password):
    users = load_users()
    users[username] = hash_password(new_password)
    save_users(users)


# ------------------------- Helper: Search Symbol -------------------------
def search_symbol(query):
    url = f"https://query1.finance.yahoo.com/v1/finance/search?q={query}"
    response = requests.get(url)
    if response.status_code == 200:
        data = response.json()
        if "quotes" in data and len(data["quotes"]) > 0:
            return data["quotes"][0]["symbol"]
    return None


# ------------------------- Session Init -------------------------
if "page" not in st.session_state:
    st.session_state.page = "login"
if "username" not in st.session_state:
    st.session_state.username = ""
if "logged_in" not in st.session_state:
    st.session_state.logged_in = False
```

```
# ------------------------- Login UI -------------------------
if st.session_state.page == "login":
    st.markdown("## 📲 OG 11 Navigation")
    st.markdown("### Select Option")
    login_opt = st.radio("", ["🟢 Login", "⚪ Sign Up"], horizontal=True)

    if login_opt == "🟢 Login":
        st.markdown("### 🔐 Login to Your Account")
        col1, col2, col3 = st.columns([1, 2, 1])
        with col2:
            username = st.text_input("📱 Enter Mobile Number", max_chars=10)
            password = st.text_input("🔑 Enter 4-digit Password", type="password", max_chars=4)
            if st.button("Login"):
                if check_login(username, password):
                    st.session_state.logged_in = True
                    st.session_state.username = username
                    st.session_state.page = "dashboard"
                    st.experimental_rerun()
                else:
                    st.error("Invalid mobile number or password")

    else:
        st.markdown("### 📝 Create a New Account")
        col1, col2, col3 = st.columns([1, 2, 1])
        with col2:
            new_user = st.text_input("👤 Choose a Mobile Number", max_chars=10)
            new_pin = st.text_input("🔑 Choose 4-digit Password", type="password", max_chars=4)
            confirm_pin = st.text_input("🔄 Confirm Password", type="password", max_chars=4)
            if st.button("Sign Up"):
                if not new_user or not new_pin or not confirm_pin:
                    st.error("All fields are required")
                elif not new_pin.isdigit() or len(new_pin) != 4:
                    st.error("Password must be exactly 4 digits")
                elif new_pin != confirm_pin:
                    st.error("Passwords do not match")
                elif signup_user(new_user, new_pin):
                    st.success("Account created successfully! Please log in.")
                    st.session_state.page = "login"
                else:
                    st.error("User already exists")

# ------------------------- Dashboard -------------------------
elif st.session_state.page == "dashboard" and st.session_state.logged_in:
    st.sidebar.markdown(f"👤 **{st.session_state.username}**")
    if st.sidebar.button("🔑 Change Password"):
        st.session_state.page = "change"
        st.experimental_rerun()
    if st.sidebar.button("🚪 Logout"):
        st.session_state.page = "login"
        st.session_state.logged_in = False
        st.experimental_rerun()

    st.title("📈 Stock Price Prediction with LSTM")

    company_query = st.text_input("Enter Company Name (e.g., Apple, Infosys, Tesla)", value="Apple")

    if st.button("Search & Predict"):
        symbol = search_symbol(company_query)
        if not symbol:
            st.error("Could not find stock symbol for that company name.")
        else:
            st.success(f"Found symbol: {symbol}")

            profile = USER_PROFILES.get(company_query, None)
            if profile:
                st.markdown(f"**Company Name:** {profile['name']}")
                st.markdown(f"**Age:** {profile['age']}")
                st.markdown(f"**Contact:** {profile['number']}")

            with st.spinner("Fetching data..."):
                df = yf.download(symbol, start="2015-01-01", end="2023-12-31")

            if df.empty:
```

```
    st.error("No data found")
else:
    df = df[["Close"]].dropna()
    scaler = MinMaxScaler()
    scaled = scaler.fit_transform(df)

    X, y = [], []
    for i in range(60, len(scaled)):
        X.append(scaled[i-60:i, 0])
        y.append(scaled[i, 0])
    X, y = np.array(X), np.array(y)
    X = X.reshape((X.shape[0], X.shape[1], 1))

    model = Sequential()
    model.add(Input(shape=(60,1)))
    model.add(LSTM(50, return_sequences=True))
    model.add(LSTM(50))
    model.add(Dense(1))
    model.compile(optimizer='adam', loss='mean_squared_error')

    with st.spinner("Training model..."):
        model.fit(X, y, epochs=3, batch_size=32, verbose=0)
        model.save("model.h5")

    pred = model.predict(X)
```

```
pred_price = scaler.inverse_transform(pred)
actual_price = scaler.inverse_transform(y.reshape(-1, 1))

st.subheader("📊 Actual vs Predicted")
fig1, ax1 = plt.subplots()
ax1.plot(actual_price, label="Actual")
ax1.plot(pred_price, label="Predicted")
ax1.legend()
st.pyplot(fig1)
plt.savefig("actual_vs_predicted.png")

# Technical Indicators
st.subheader("📉 Technical Indicators")
df['MA_14'] = df['Close'].rolling(window=14).mean()

def rsi(series, period=14):
    delta = series.diff()
    gain = delta.clip(lower=0)
    loss = -delta.clip(upper=0)
    avg_gain = gain.rolling(window=period).mean()
    avg_loss = loss.rolling(window=period).mean()
    rs = avg_gain / avg_loss
    return 100 - (100 / (1 + rs))

df['RSI_14'] = rsi(df['Close'])
```

```
                fig2, (a1, a2) = plt.subplots(2, 1, figsize=(10, 6))
                a1.plot(df['Close'], label='Close')
                a1.plot(df['MA_14'], label='MA 14')
                a1.legend()
                a2.plot(df['RSI_14'], label='RSI 14', color='purple')
                a2.axhline(70, color='red', linestyle='--')
                a2.axhline(30, color='green', linestyle='--')
                a2.legend()
                st.pyplot(fig2)
                plt.savefig("ma_rsi_plot.png")

# -------------------------- Change Password Page --------------------------
elif st.session_state.page == "change":
    st.title("🔑 Change Password")
    old = st.text_input("Current PIN", type="password", max_chars=4)
    new = st.text_input("New 4-digit PIN", type="password", max_chars=4)
    confirm = st.text_input("Confirm New PIN", type="password", max_chars=4)

    if st.button("Update Password"):
        if not check_login(st.session_state.username, old):
            st.error("Incorrect current PIN")
        elif new != confirm:
            st.error("PINs do not match")
        else:
            change_password(st.session_state.username, new)
            st.success("Password changed successfully")
            st.session_state.page = "dashboard"

else:
    st.warning("Please login to use the app.")
```