#!/usr/bin/env python3

"""

SportAI Suite Enterprise Edition - Setup Script

This script initializes the application for first-time use

"""

import os

import sys

import json

import secrets

import sqlite3

import subprocess

from pathlib import Path

from datetime import datetime

import argparse

import getpass

import hashlib

# Colors for terminal output

class Colors:

HEADER = '\033[95m'

OKBLUE = '\033[94m'

OKCYAN = '\033[96m'

OKGREEN = '\033[92m'

WARNING = '\033[93m'

FAIL = '\033[91m'

ENDC = '\033[0m'

BOLD = '\033[1m'

UNDERLINE = '\033[4m'

def print\_header():

"""Print application header"""

print(f"""

{Colors.OKBLUE}╔══════════════════════════════════════════════════════════════╗

║ ║

║ {Colors.BOLD}🏟️ SportAI Suite Enterprise Edition - Setup Wizard{Colors.ENDC}{Colors.OKBLUE} ║

║ ║

║ Version: 3.0.0 ║

║ Copyright (c) 2025 SportAI Technologies ║

║ ║

╚══════════════════════════════════════════════════════════════╝{Colors.ENDC}

""")

def check\_python\_version():

"""Check if Python version meets requirements"""

print(f"\n{Colors.OKCYAN}🔍 Checking Python version...{Colors.ENDC}")

if sys.version\_info < (3, 8):

print(f"{Colors.FAIL}❌ Python 3.8 or higher is required. You have {sys.version}{Colors.ENDC}")

sys.exit(1)

print(f"{Colors.OKGREEN}✅ Python {sys.version} is compatible{Colors.ENDC}")

def create\_directories():

"""Create required directories"""

print(f"\n{Colors.OKCYAN}📁 Creating directory structure...{Colors.ENDC}")

directories = [

'database',

'logs',

'audit\_logs',

'configurations',

'uploads',

'backups',

'static',

'ai\_modules',

'modules/facility\_management',

'modules/membership\_management',

'modules/event\_management',

'modules/financial\_management',

'modules/reporting',

'tests',

'docs',

'scripts'

]

for directory in directories:

Path(directory).mkdir(parents=True, exist\_ok=True)

print(f" ✅ Created {directory}/")

# Create .gitkeep files

for directory in directories:

gitkeep = Path(directory) / '.gitkeep'

gitkeep.touch()

def install\_dependencies():

"""Install Python dependencies"""

print(f"\n{Colors.OKCYAN}📦 Installing dependencies...{Colors.ENDC}")

try:

# Upgrade pip

subprocess.check\_call([sys.executable, "-m", "pip", "install", "--upgrade", "pip"])

# Install requirements

if Path("requirements.txt").exists():

subprocess.check\_call([sys.executable, "-m", "pip", "install", "-r", "requirements.txt"])

print(f"{Colors.OKGREEN}✅ All dependencies installed successfully{Colors.ENDC}")

else:

print(f"{Colors.WARNING}⚠️ requirements.txt not found. Installing core dependencies...{Colors.ENDC}")

core\_deps = ["streamlit", "pandas", "numpy", "plotly", "sqlalchemy", "bcrypt", "python-dotenv"]

subprocess.check\_call([sys.executable, "-m", "pip", "install"] + core\_deps)

except subprocess.CalledProcessError as e:

print(f"{Colors.FAIL}❌ Failed to install dependencies: {e}{Colors.ENDC}")

print(f"{Colors.WARNING}Please install manually: pip install -r requirements.txt{Colors.ENDC}")

def setup\_environment():

"""Set up environment variables"""

print(f"\n{Colors.OKCYAN}🔧 Setting up environment configuration...{Colors.ENDC}")

if Path(".env").exists():

print(f"{Colors.WARNING}⚠️ .env file already exists. Skipping...{Colors.ENDC}")

return

# Copy from template if exists

if Path(".env.example").exists():

with open(".env.example", 'r') as src, open(".env", 'w') as dst:

content = src.read()

# Generate secure keys

content = content.replace("your-very-secure-secret-key-here-change-this",

secrets.token\_urlsafe(32))

content = content.replace("your-license-key-here",

f"TRIAL-{secrets.token\_hex(16).upper()}")

dst.write(content)

print(f"{Colors.OKGREEN}✅ Environment configuration created{Colors.ENDC}")

else:

# Create minimal .env

env\_content = f"""

# SportAI Suite Configuration

APP\_NAME=SportAI Suite Enterprise

APP\_ENV=production

SECRET\_KEY={secrets.token\_urlsafe(32)}

DATABASE\_URL=sqlite:///database/sportai.db

SESSION\_TIMEOUT=3600

LICENSE\_KEY=TRIAL-{secrets.token\_hex(16).upper()}

"""

with open(".env", 'w') as f:

f.write(env\_content)

print(f"{Colors.OKGREEN}✅ Basic environment configuration created{Colors.ENDC}")

def setup\_database():

"""Initialize database"""

print(f"\n{Colors.OKCYAN}🗄️ Setting up database...{Colors.ENDC}")

db\_path = Path("database/sportai.db")

if db\_path.exists():

response = input(f"{Colors.WARNING}Database already exists. Reset? (y/N): {Colors.ENDC}")

if response.lower() != 'y':

print("Keeping existing database.")

return

# Create database

conn = sqlite3.connect(str(db\_path))

cursor = conn.cursor()

# Create tables

cursor.execute("""

CREATE TABLE IF NOT EXISTS users (

id INTEGER PRIMARY KEY AUTOINCREMENT,

email TEXT UNIQUE NOT NULL,

password\_hash TEXT NOT NULL,

role TEXT NOT NULL,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

last\_login TIMESTAMP,

failed\_attempts INTEGER DEFAULT 0,

locked\_until TIMESTAMP,

must\_change\_password BOOLEAN DEFAULT 0,

two\_factor\_enabled BOOLEAN DEFAULT 0,

api\_key TEXT UNIQUE

)

""")

cursor.execute("""

CREATE TABLE IF NOT EXISTS sessions (

id TEXT PRIMARY KEY,

user\_id INTEGER,

token TEXT UNIQUE NOT NULL,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

expires\_at TIMESTAMP,

FOREIGN KEY (user\_id) REFERENCES users (id)

)

""")

cursor.execute("""

CREATE TABLE IF NOT EXISTS audit\_logs (

id INTEGER PRIMARY KEY AUTOINCREMENT,

timestamp TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

user\_email TEXT,

action TEXT,

details TEXT,

ip\_address TEXT,

session\_id TEXT

)

""")

conn.commit()

conn.close()

print(f"{Colors.OKGREEN}✅ Database initialized{Colors.ENDC}")

def create\_admin\_user():

"""Create initial admin user"""

print(f"\n{Colors.OKCYAN}👤 Creating admin user...{Colors.ENDC}")

email = input(f"{Colors.BOLD}Admin email [admin@sportai.com]: {Colors.ENDC}") or "admin@sportai.com"

# Get password securely

while True:

password = getpass.getpass(f"{Colors.BOLD}Admin password (min 8 chars): {Colors.ENDC}")

if len(password) < 8:

print(f"{Colors.FAIL}Password must be at least 8 characters{Colors.ENDC}")

continue

confirm = getpass.getpass(f"{Colors.BOLD}Confirm password: {Colors.ENDC}")

if password != confirm:

print(f"{Colors.FAIL}Passwords do not match{Colors.ENDC}")

continue

break

# Hash password

salt = secrets.token\_hex(32)

pepper = "SportAI\_Secure\_2025"

combined = f"{password}{salt}{pepper}"

password\_hash = hashlib.pbkdf2\_hmac('sha256', combined.encode(), salt.encode(), 100000).hex()

# Save to database

db\_path = Path("database/sportai.db")

conn = sqlite3.connect(str(db\_path))

cursor = conn.cursor()

try:

cursor.execute("""

INSERT INTO users (email, password\_hash, role, api\_key, must\_change\_password)

VALUES (?, ?, 'admin', ?, 0)

""", (email, password\_hash, secrets.token\_urlsafe(32)))

conn.commit()

print(f"{Colors.OKGREEN}✅ Admin user created: {email}{Colors.ENDC}")

except sqlite3.IntegrityError:

print(f"{Colors.WARNING}⚠️ User {email} already exists{Colors.ENDC}")

finally:

conn.close()

# Save salt

with open(".salt", 'w') as f:

f.write(salt)

def setup\_ssl\_certificates():

"""Generate self-signed SSL certificates for development"""

print(f"\n{Colors.OKCYAN}🔒 Setting up SSL certificates...{Colors.ENDC}")

ssl\_dir = Path("nginx/ssl")

ssl\_dir.mkdir(parents=True, exist\_ok=True)

if (ssl\_dir / "cert.pem").exists():

print(f"{Colors.WARNING}⚠️ SSL certificates already exist. Skipping...{Colors.ENDC}")

return

try:

# Generate self-signed certificate

subprocess.run([

"openssl", "req", "-x509", "-newkey", "rsa:4096",

"-keyout", str(ssl\_dir / "key.pem"),

"-out", str(ssl\_dir / "cert.pem"),

"-days", "365", "-nodes",

"-subj", "/C=US/ST=State/L=City/O=SportAI/CN=localhost"

], check=True, capture\_output=True)

print(f"{Colors.OKGREEN}✅ SSL certificates generated{Colors.ENDC}")

except (subprocess.CalledProcessError, FileNotFoundError):

print(f"{Colors.WARNING}⚠️ Could not generate SSL certificates. OpenSSL may not be installed.{Colors.ENDC}")

def create\_sample\_data():

"""Create sample data for testing"""

response = input(f"\n{Colors.OKCYAN}📊 Create sample data for testing? (y/N): {Colors.ENDC}")

if response.lower() != 'y':

return

print(f"{Colors.OKCYAN}Creating sample data...{Colors.ENDC}")

# Create sample configuration

config = {

"facility": {

"name": "Demo Sports Complex",

"type": "multi-sport",

"timezone": "America/Chicago"

},

"subscription": {

"tier": "professional",

"valid\_until": "2025-12-31T23:59:59"

}

}

with open("configurations/demo\_config.json", 'w') as f:

json.dump(config, f, indent=2)

print(f"{Colors.OKGREEN}✅ Sample data created{Colors.ENDC}")

def run\_tests():

"""Run basic tests to verify installation"""

print(f"\n{Colors.OKCYAN}🧪 Running installation tests...{Colors.ENDC}")

tests\_passed = True

# Test imports

try:

import streamlit

print(f" ✅ Streamlit imported successfully")

except ImportError:

print(f" {Colors.FAIL}❌ Failed to import Streamlit{Colors.ENDC}")

tests\_passed = False

# Test database connection

try:

conn = sqlite3.connect("database/sportai.db")

conn.close()

print(f" ✅ Database connection successful")

except Exception as e:

print(f" {Colors.FAIL}❌ Database connection failed: {e}{Colors.ENDC}")

tests\_passed = False

# Test file permissions

try:

test\_file = Path("logs/test.tmp")

test\_file.touch()

test\_file.unlink()

print(f" ✅ File permissions OK")

except Exception as e:

print(f" {Colors.FAIL}❌ File permission issue: {e}{Colors.ENDC}")

tests\_passed = False

if tests\_passed:

print(f"{Colors.OKGREEN}✅ All tests passed!{Colors.ENDC}")

else:

print(f"{Colors.WARNING}⚠️ Some tests failed. Please check the errors above.{Colors.ENDC}")

def print\_next\_steps():

"""Print next steps for the user"""

print(f"""

{Colors.OKBLUE}╔══════════════════════════════════════════════════════════════╗

║ ║

║ {Colors.BOLD}🎉 Setup Complete!{Colors.ENDC}{Colors.OKBLUE} ║

║ ║

╚══════════════════════════════════════════════════════════════╝{Colors.ENDC}

{Colors.OKGREEN}Next Steps:{Colors.ENDC}

1. {Colors.BOLD}Start the application:{Colors.ENDC}

streamlit run sportai\_main\_app\_file.py

2. {Colors.BOLD}Access the application:{Colors.ENDC}

http://localhost:8501

3. {Colors.BOLD}Login with your admin credentials{Colors.ENDC}

4. {Colors.BOLD}Configure your facility:{Colors.ENDC}

Settings → Configuration → Facility Information

5. {Colors.BOLD}Add users:{Colors.ENDC}

Settings → Users → Add New User

{Colors.OKCYAN}For production deployment:{Colors.ENDC}

docker-compose up -d

{Colors.WARNING}Need help?{Colors.ENDC}

Documentation: https://docs.sportai.com

Support: support@sportai.com

{Colors.OKGREEN}Thank you for choosing SportAI Suite!{Colors.ENDC}

""")

def main():

"""Main setup function"""

parser = argparse.ArgumentParser(description='SportAI Suite Setup Wizard')

parser.add\_argument('--skip-deps', action='store\_true', help='Skip dependency installation')

parser.add\_argument('--reset', action='store\_true', help='Reset existing installation')

parser.add\_argument('--quiet', action='store\_true', help='Minimal output')

args = parser.parse\_args()

print\_header()

if args.reset:

response = input(f"{Colors.WARNING}⚠️ This will reset your installation. Continue? (y/N): {Colors.ENDC}")

if response.lower() != 'y':

print("Setup cancelled.")

return

# Run setup steps

check\_python\_version()

create\_directories()

if not args.skip\_deps:

install\_dependencies()

setup\_environment()

setup\_database()

create\_admin\_user()

setup\_ssl\_certificates()

create\_sample\_data()

run\_tests()

print\_next\_steps()

if \_\_name\_\_ == "\_\_main\_\_":

try:

main()

except KeyboardInterrupt:

print(f"\n{Colors.WARNING}Setup interrupted by user{Colors.ENDC}")

sys.exit(1)

except Exception as e:

print(f"\n{Colors.FAIL}Setup failed: {e}{Colors.ENDC}")

sys.exit(1)