# create\_users\_table.py

"""

ScoutConnect - Users Table Creation Script

Creates the users table with authentication and role-based access control support.

"""

import sqlite3

import hashlib

from datetime import datetime

from enum import Enum

class UserRole(Enum):

"""User roles for role-based access control."""

ADMIN = "admin"

SCOUT = "scout"

COACH = "coach"

ORGANIZATION = "organization"

VIEWER = "viewer"

def create\_users\_table():

"""Create the users table with all necessary fields for ScoutConnect."""

# Connect to database

conn = sqlite3.connect('scoutconnect.db')

cursor = conn.cursor()

print("🔨 Creating users table...")

# Drop table if it exists (for development)

cursor.execute("DROP TABLE IF EXISTS users")

# Create users table

cursor.execute("""

CREATE TABLE users (

id INTEGER PRIMARY KEY AUTOINCREMENT,

-- Authentication fields

username VARCHAR(50) UNIQUE NOT NULL,

email VARCHAR(100) UNIQUE NOT NULL,

password\_hash VARCHAR(255) NOT NULL,

salt VARCHAR(32) NOT NULL,

-- Profile information

first\_name VARCHAR(50) NOT NULL,

last\_name VARCHAR(50) NOT NULL,

display\_name VARCHAR(100),

-- Role and permissions

role VARCHAR(20) NOT NULL DEFAULT 'viewer',

is\_active BOOLEAN DEFAULT 1,

is\_verified BOOLEAN DEFAULT 0,

-- Organization/Team affiliation

organization\_name VARCHAR(100),

team\_name VARCHAR(100),

position\_title VARCHAR(100),

-- Contact information

phone VARCHAR(20),

location VARCHAR(100),

bio TEXT,

-- Professional details

years\_experience INTEGER,

specialization TEXT, -- JSON field for sports/positions they specialize in

certifications TEXT, -- JSON field for certifications

-- Account metadata

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

last\_login TIMESTAMP,

login\_attempts INTEGER DEFAULT 0,

locked\_until TIMESTAMP,

-- Settings and preferences

preferences TEXT, -- JSON field for user preferences

notification\_settings TEXT, -- JSON field for notification preferences

timezone VARCHAR(50) DEFAULT 'UTC',

-- Security

email\_verification\_token VARCHAR(255),

password\_reset\_token VARCHAR(255),

password\_reset\_expires TIMESTAMP,

two\_factor\_enabled BOOLEAN DEFAULT 0,

two\_factor\_secret VARCHAR(32),

-- Check constraints

CHECK (role IN ('admin', 'scout', 'coach', 'organization', 'viewer')),

CHECK (years\_experience >= 0),

CHECK (login\_attempts >= 0),

CHECK (email LIKE '%@%')

)

""")

# Create indexes for better query performance

cursor.execute("CREATE INDEX idx\_users\_email ON users(email)")

cursor.execute("CREATE INDEX idx\_users\_username ON users(username)")

cursor.execute("CREATE INDEX idx\_users\_role ON users(role)")

cursor.execute("CREATE INDEX idx\_users\_organization ON users(organization\_name)")

cursor.execute("CREATE INDEX idx\_users\_active ON users(is\_active)")

cursor.execute("CREATE INDEX idx\_users\_created\_at ON users(created\_at)")

print("✅ Users table created successfully!")

# Create some sample users for development

create\_sample\_users(cursor)

# Commit changes

conn.commit()

conn.close()

print("💾 Database changes committed!")

def hash\_password(password: str, salt: str = None) -> tuple:

"""Hash password with salt for secure storage."""

if salt is None:

salt = hashlib.sha256(str(datetime.now()).encode()).hexdigest()[:32]

password\_hash = hashlib.pbkdf2\_hmac(

'sha256',

password.encode('utf-8'),

salt.encode('utf-8'),

100000 # iterations

).hex()

return password\_hash, salt

def create\_sample\_users(cursor):

"""Create sample users for development and testing."""

print("👥 Creating sample users...")

sample\_users = [

{

'username': 'admin\_user',

'email': 'admin@scoutconnect.dev',

'password': 'admin123',

'first\_name': 'System',

'last\_name': 'Administrator',

'role': 'admin',

'organization\_name': 'ScoutConnect',

'position\_title': 'System Administrator',

'is\_verified': 1

},

{

'username': 'scout\_mike',

'email': 'mike.scout@example.com',

'password': 'scout123',

'first\_name': 'Mike',

'last\_name': 'Johnson',

'role': 'scout',

'organization\_name': 'Elite Sports Scouting',

'position\_title': 'Senior Scout',

'years\_experience': 8,

'specialization': '["basketball", "football"]',

'location': 'Los Angeles, CA',

'is\_verified': 1

},

{

'username': 'coach\_sarah',

'email': 'sarah.coach@highschool.edu',

'password': 'coach123',

'first\_name': 'Sarah',

'last\_name': 'Williams',

'role': 'coach',

'organization\_name': 'Lincoln High School',

'team\_name': 'Lincoln Lions',

'position\_title': 'Head Basketball Coach',

'years\_experience': 12,

'specialization': '["basketball"]',

'location': 'Chicago, IL',

'is\_verified': 1

},

{

'username': 'org\_manager',

'email': 'manager@sportsorg.com',

'password': 'org123',

'first\_name': 'Robert',

'last\_name': 'Davis',

'role': 'organization',

'organization\_name': 'Metro Sports Alliance',

'position\_title': 'Talent Acquisition Manager',

'years\_experience': 5,

'location': 'New York, NY',

'is\_verified': 1

},

{

'username': 'viewer\_john',

'email': 'john.viewer@example.com',

'password': 'viewer123',

'first\_name': 'John',

'last\_name': 'Smith',

'role': 'viewer',

'position\_title': 'Sports Enthusiast',

'is\_verified': 0

}

]

for user\_data in sample\_users:

# Hash password

password\_hash, salt = hash\_password(user\_data['password'])

# Prepare insert data

cursor.execute("""

INSERT INTO users (

username, email, password\_hash, salt, first\_name, last\_name,

role, organization\_name, team\_name, position\_title,

years\_experience, specialization, location, is\_verified

) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)

""", (

user\_data['username'],

user\_data['email'],

password\_hash,

salt,

user\_data['first\_name'],

user\_data['last\_name'],

user\_data['role'],

user\_data.get('organization\_name'),

user\_data.get('team\_name'),

user\_data.get('position\_title'),

user\_data.get('years\_experience'),

user\_data.get('specialization'),

user\_data.get('location'),

user\_data.get('is\_verified', 0)

))

print(f"✅ Created {len(sample\_users)} sample users")

def verify\_users\_table():

"""Verify the users table was created correctly."""

conn = sqlite3.connect('scoutconnect.db')

cursor = conn.cursor()

print("\n🔍 Verifying users table...")

# Check table structure

cursor.execute("PRAGMA table\_info(users)")

columns = cursor.fetchall()

print(f"📊 Table has {len(columns)} columns:")

for col in columns:

print(f" - {col[1]} ({col[2]})")

# Check indexes

cursor.execute("PRAGMA index\_list(users)")

indexes = cursor.fetchall()

print(f"📈 Table has {len(indexes)} indexes")

# Count users

cursor.execute("SELECT COUNT(\*) FROM users")

user\_count = cursor.fetchone()[0]

print(f"👥 Table contains {user\_count} users")

# Show sample users

cursor.execute("""

SELECT id, username, email, role, organization\_name, is\_verified

FROM users

ORDER BY role, username

""")

users = cursor.fetchall()

print("\n📋 Sample users:")

for user in users:

verified\_status = "✓" if user[5] else "✗"

print(f" {user[0]:2d}. {user[1]:15s} | {user[3]:12s} | {user[4] or 'No org':20s} | {verified\_status}")

conn.close()

return True

def create\_user\_related\_tables():

"""Create additional tables related to users."""

conn = sqlite3.connect('scoutconnect.db')

cursor = conn.cursor()

print("\n🔗 Creating user-related tables...")

# User sessions table for JWT token management

cursor.execute("""

CREATE TABLE IF NOT EXISTS user\_sessions (

id INTEGER PRIMARY KEY AUTOINCREMENT,

user\_id INTEGER NOT NULL,

token\_jti VARCHAR(36) UNIQUE NOT NULL, -- JWT ID

device\_info TEXT,

ip\_address VARCHAR(45),

user\_agent TEXT,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

expires\_at TIMESTAMP NOT NULL,

is\_revoked BOOLEAN DEFAULT 0,

FOREIGN KEY (user\_id) REFERENCES users(id) ON DELETE CASCADE

)

""")

cursor.execute("CREATE INDEX idx\_sessions\_user\_id ON user\_sessions(user\_id)")

cursor.execute("CREATE INDEX idx\_sessions\_token ON user\_sessions(token\_jti)")

cursor.execute("CREATE INDEX idx\_sessions\_expires ON user\_sessions(expires\_at)")

# User activity log

cursor.execute("""

CREATE TABLE IF NOT EXISTS user\_activity\_log (

id INTEGER PRIMARY KEY AUTOINCREMENT,

user\_id INTEGER NOT NULL,

action VARCHAR(50) NOT NULL,

resource\_type VARCHAR(50),

resource\_id INTEGER,

ip\_address VARCHAR(45),

user\_agent TEXT,

timestamp TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

details TEXT, -- JSON field for additional details

FOREIGN KEY (user\_id) REFERENCES users(id) ON DELETE CASCADE

)

""")

cursor.execute("CREATE INDEX idx\_activity\_user\_id ON user\_activity\_log(user\_id)")

cursor.execute("CREATE INDEX idx\_activity\_timestamp ON user\_activity\_log(timestamp)")

cursor.execute("CREATE INDEX idx\_activity\_action ON user\_activity\_log(action)")

conn.commit()

conn.close()

print("✅ User-related tables created successfully!")

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

print("🚀 ScoutConnect Users Table Setup")

print("=" \* 50)

try:

# Create main users table

create\_users\_table()

# Create related tables

create\_user\_related\_tables()

# Verify everything was created correctly

verify\_users\_table()

print("\n🎉 Users table setup complete!")

print("\n📝 Next steps:")

print("1. Create SQLAlchemy models for the users table")

print("2. Implement authentication endpoints")

print("3. Add password hashing with bcrypt")

print("4. Create JWT token generation/validation")

print("5. Add role-based access control decorators")

except Exception as e:

print(f"❌ Error setting up users table: {e}")

raise