CODING:

```
from flask import Flask, render_template, request, redirect, url_for, session, flash
from flask_sqlalchemy import SQLAlchemy

app = Flask(__name__)
app.secret_key = 'your_secret_key'
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///database.db'
db = SQLAlchemy(app)

# Database model
class User(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(150))
    location = db.Column(db.String(150))
    age = db.Column(db.Integer)
    mobile = db.Column(db.String(20))
    username = db.Column(db.String(150), unique=True)
    password = db.Column(db.String(150))

# Home page
@app.route('/')
def index():
    return render_template('index.html')

# Register
@app.route('/register', methods=['GET', 'POST'])
def register():
    if request.method == 'POST':
        name = request.form['name']
        location = request.form['location']
        age = request.form['age']
        mobile = request.form['mobile']
        username = request.form['username']
        password = request.form['password']

        existing_user = User.query.filter_by(username=username).first()
        if existing_user:
            flash('Username already exists!')
            return redirect(url_for('register'))

        new_user = User(name=name, location=location, age=age, mobile=mobile,
username=username, password=password)
        db.session.add(new_user)
        db.session.commit()
        flash('Registration successful. Please log in.')
        return redirect(url_for('login'))
    return render_template('register.html')

# Login
@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']
        user = User.query.filter_by(username=username, password=password).first()
        if user:
            session['user_id'] = user.id
            return redirect("https://huggingface.co/spaces/josephchay/Soundsation")
        else:
            flash('Invalid credentials. Try again.')
            return redirect(url_for('login'))
    return render_template('login.html')

if __name__ == '__main__':
    with app.app_context():
        db.create_all()
    app.run(debug=True,port=5001)
from flask import Flask, request, render_template, send_file
import google.generativeai as genai
from gtts import gTTS
import os
import logging
import uuid

app = Flask(__name__)

# Configure logging
logging.basicConfig(level=logging.INFO)
logger = logging.getLogger(__name__)

# Configure environment variables
GOOGLE_API_KEY = os.getenv("GOOGLE_API_KEY")  # Gemini API key

# Configure Gemini API
try:
    genai.configure(api_key=GOOGLE_API_KEY)
    model = genai.GenerativeModel("gemini-2.0-flash")
    logger.info("Gemini API configured successfully")
except Exception as e:
    logger.error(f"Failed to configure Gemini API: {e}")
    raise

# Ensure static directory exists
if not os.path.exists("static"):
    os.makedirs("static")

@app.route("/", methods=["GET", "POST"])
def index():
    if request.method == "POST":
        # Get form data
        cinematic_theme = request.form.get("cinematic_theme", "epic adventure like
The Lord of the Rings")
        song_type = request.form.get("song_type", "pop")
        language = request.form.get("language", "english")
        voice_gender = request.form.get("voice_gender", "female")

        # Generate lyrics
        try:
            prompt = f"""
            Generate song lyrics inspired by a cinematic theme: {cinematic_theme}.
            The song should be in the {song_type} genre, with an uplifting and heroic
tone.
            Write the lyrics in {language.capitalize()} (ensure correct grammar and
cultural relevance for {language}).
            Structure the song with a verse, chorus, and bridge.
            """
            response = model.generate_content(prompt)
            lyrics = response.text
            logger.info(f"Lyrics generated successfully in {language}")
        except Exception as e:
            logger.error(f"Failed to generate lyrics: {e}")
            return render_template("index.html", error=f"Error generating lyrics: {e}")

        # Convert lyrics to MP3 audio using gTTS
        try:
            tts_language = "ta" if language.lower() == "tamil" else "en"
            tts = gTTS(text=lyrics, lang=tts_language, slow=False)
            audio_path = f"static/lyrics_song_{uuid.uuid4()}.mp3"
            tts.save(audio_path)
            logger.info(f"Audio generated successfully for {language} (gTTS)")
        except Exception as e:
            logger.error(f"Failed to generate audio: {e}")
            return render_template("index.html", error=f"Error generating audio: {e}")

        return render_template(
            "index.html",
            lyrics=lyrics,
            audio_file=f"/{audio_path}",
            cinematic_theme=cinematic_theme,
            song_type=song_type,
            language=language,
            voice_gender=voice_gender
        )

    return render_template("index.html", lyrics=None, audio_file=None)
```

```
@app.route("/download/<path:filename>")
def download_file(filename):
    try:
        return send_file(filename, as_attachment=True)
    except Exception as e:
        logger.error(f"Failed to serve file {filename}: {e}")
        return f"Error downloading file: {e}", 500


if __name__ == "__main__":
    app.run(debug=True)
```