```
from flask import Flask, request, jsonify, render_template
from flask_sqlalchemy import SQLAlchemy
from datetime import datetime
import os

app = Flask(__name__)

# Ensure the instance folder exists
if not os.path.exists('instance'):
    os.makedirs('instance')

# Configure SQLite database with absolute path
db_path = os.path.join(os.path.abspath(os.path.dirname(__file__)), 'instance',
'apps.db')
app.config['SQLALCHEMY_DATABASE_URI'] = f'sqlite:///{db_path}'
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
db = SQLAlchemy(app)

# App Model
class App(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    app_name = db.Column(db.String(100), nullable=False)
    version = db.Column(db.String(20), nullable=False)
    description = db.Column(db.Text, nullable=True)
    created_at = db.Column(db.DateTime, default=datetime.utcnow)

    def to_dict(self):
        return {
            'id': self.id,
            'app_name': self.app_name,
            'version': self.version,
            'description': self.description,
            'created_at': self.created_at.isoformat()
        }

# Create the database tables
with app.app_context():
    db.create_all()
    print(f"Database created at: {db_path}")

@app.route('/')
def home():
    return render_template('home.html')
```

```
@app.route('/add-app', methods=['POST'])
def add_app():
    try:
        data = request.get_json() if request.is_json else request.form

        if not all(key in data for key in ['app_name', 'version']):
            return jsonify({'error': 'Missing required fields'}), 400

        new_app = App(
            app_name=data['app_name'],
            version=data['version'],
            description=data.get('description', '')
        )

        db.session.add(new_app)
        db.session.commit()

        print(f"Added new app to database: {new_app.app_name}")

        return jsonify({
            'message': 'App added successfully',
            'app': new_app.to_dict()
        }), 201

    except Exception as e:
        db.session.rollback()
        print(f"Error adding app: {str(e)}")
        return jsonify({'error': str(e)}), 500

@app.route('/get-all-apps', methods=['GET'])
def get_all_apps():
    try:
        apps = App.query.all()
        return jsonify([app.to_dict() for app in apps])
    except Exception as e:
        return jsonify({'error': str(e)}), 500

@app.route('/get-app/<int:id>', methods=['GET'])
def get_app(id):
    try:
        app = App.query.get(id)
        if app is None:
            return jsonify({'error': 'App not found'}), 404
        return jsonify(app.to_dict()), 200

    except Exception as e:
        return jsonify({'error': str(e)}), 500

@app.route('/delete-app/<int:id>', methods=['DELETE'])
def delete_app(id):
    try:
        app = App.query.get(id)
        if app is None:
            return jsonify({'error': 'App not found'}), 404

        db.session.delete(app)
        db.session.commit()

        print(f"Deleted app from database: {app.app_name}")

        return jsonify({'message': 'App deleted successfully'}), 200

    except Exception as e:
        db.session.rollback()
        print(f"Error deleting app: {str(e)}")
        return jsonify({'error': str(e)}), 500

@app.route('/view-database')
def view_database():
    try:
        apps = App.query.all()
        return render_template('database.html', apps=apps)
    except Exception as e:
        return f"Error: {str(e)}"

if __name__ == '__main__':
    app.run(debug=True, port=5000)
```