2024.09.22
Music Store System Installation and Configuration Guide, v1

Welcome to the Music Store System installation and configuration guide.
This document will help you set up the Music Store application on Windows 10 or 11.
The guide is designed for IT students with basic knowledge of Flask, Python, HTML, and databases.

Index:

1. Introduction
The Music Store System is a web application built using Flask, a web framework for Python.
This guide will walk you through the steps to install and configure the application on a Windows environment.

2. Prerequisites
Before you begin, ensure you have the following installed on your system:

Python 3.8 or newer
PostgreSQL
Git for cloning the repository

3. Installing Python and Flask

Install Python:
Download the latest version of Python from the official website.
Run the installer and ensure you check the option to add Python to your system PATH.

Install Flask:
Open Command Prompt-tool in Windows and create a virtual environment:
mkdir myproject
cd myproject
python -m venv venv

Activate the virtual environment:
venv\Scripts\activate

Install Flask:
pip install Flask

4. Setting Up the Music Store Application

If you have Git installed, you can clone the repository:
git clone https://github.com/tihyyti/MusicStore.git
cd music_store (your local repo in your PC)

Install Dependencies:
Ensure your virtual environment is activated and install the required packages:
pip install -r requirements.txt

5. Configuring PostgreSQL Database
Install PostgreSQL:
Download and install PostgreSQL from the official

website.
Create a Database:
Open pgAdmin and use the SQL-request tool there to create a new database:
CREATE DATABASE MStore_v1;
Create the schema mstore_v1 with pgAdmin.

Set Up the Database Schema:
Define the search path for non-public schemas:
ALTER DATABASE MStore_v1 SET search_path TO public, mstore_v1;

6. Creating and Populating the Database

Create Tables:
Use the provided SQL scripts in your local repository to create
the necessary tables in the MStore_v1 schema:
With pgAdmin-tool load the db creation script (cloned from GIT)
from your local repository and run it.
Check that 6 tables with columns and primary and foreign keys are created
(from left side menu and under mstore_v1 schema).

Populate with Test Data:

Insert test data (using pgAdmin sql-inquiries) into the tables using the SQL script provided
or a database management tool.

7. Running the Application

Set Up Environment Variables:
Create a .env file in the project root with the following content:

```
FLASK_APP=run.py
FLASK_ENV=development
DATABASE_URL=postgresql://username:password@localhost/M
Store_v1
```

Edit the app/config.py file according to you local PostgreSQL db parameters:

```
# app/config.py
imports here:
...
class Config:
    SQLALCHEMY_DATABASE_URI = os.getenv('DATABASE_URL',

'postgresql://postgres:password@localhost:5432/MStore_v
1')
    SQLALCHEMY_TRACK_MODIFICATIONS = False
    ENVIRONMENT = "development"
    FLASK_APP = "app"
    FLASK_DEBUG = True
    SECRET_KEY = "your secret key"
```

Edit the your PostgreSQL db MStore_v1 password here and calculate a secret key for your system and edit it in Class Config.
(ask from e.g. MS copilot how to calculate the secret key)

Run the Application with BASH-terminal:
CD ...yourLocalRepo/app/python3 run.py

8. Directory Structure (under construction)

```
music-store/
│
├── app/
│   ├── __init__.py
```

```
│    ├── routes/
│    │    ├── home.py
│    │    ├── customer_dashboard.py
│    │    ├── store_manager.py
│    │
│    └── templates/
│         ├── home.html
│         ├── custodashboard.html
│         ├── manager_dashboard.html
│         ├── customer_shopping_carts.html
│
├── static/
│    ├── css/
│    ├── js/
│
├── .env (optional?)
├---config.py (needed)
├── requirements.txt
├── run.py
└── README.md
```

9. User Interface Overview

The Music Store application consists of several key components:
Navigation: Provides links to different sections of the application.
Dashboards: Displays key metrics and information for customers and store managers.
Shopping Module: Allows customers to view and manage their shopping carts.