Content Curator System - Installation Guide

Version: 1.0

Date: July 2025

Target Python Version: 3.12

Table of Contents

System Overview

The Content Curator System is a Flask-based web application that provides:

- **Content Curation:** RSS feed processing and article analysis
- **AI Analysis:** Current events analysis using Ollama's Granite model
- **Customer Management:** Customer insights and data processing
- **Podcast Generation:** Audio content creation with FFmpeg
- **Research Tools:** Academic paper processing and PDF parsing
- **Interactive Chat:** Enhanced chat features with database integration

Key Technologies:

- Flask web framework
- SQLite3 database
- Ollama AI platform with Granite 3.2:8b model
- FFmpeg for audio processing
- D3.js for data visualizations
- Python 3.12 with virtual environment

Prerequisites

Before beginning installation, ensure you have:

- **Administrator/sudo access** on your system
- **Stable internet connection** (for downloading dependencies)
- **At least 4GB free disk space**
- **Project zip file** containing all source code and folders

Minimum System Requirements:

- RAM: 8GB (16GB recommended for Ollama)
- CPU: 64-bit processor
- Disk Space: 4GB free space minimum

Step 1: Python 3.12 Installation

macOS Installation

Option A: Using Homebrew (Recommended)

```
bash
```

```
# Install Homebrew if not already installed
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"

# Install Python 3.12
brew install python@3.12

# Verify installation
python3.12 --version
```

Option B: Using Official Python Installer

1. Download Python 3.12 from https://www.python.org/downloads/macos/
2. Run the `.pkg` installer
3. Follow installation wizard
4. Verify: `python3.12 --version`

Windows Installation

Using Official Python Installer (Recommended)

1. Download Python 3.12 from https://www.python.org/downloads/windows/
2. **IMPORTANT:** Check "Add Python to PATH" during installation
3. Choose "Install for all users" if you have admin rights
4. Select "Add Python to environment variables"
5. Run installer as Administrator
6. Verify installation:

```cmd
python --version
pip --version
```

Using Windows Package Manager (winget)

```cmd
# Run as Administrator
winget install Python.Python.3.12
```

Linux Installation

Ubuntu/Debian:

```bash
# Update package list
sudo apt update

# Install Python 3.12
sudo apt install python3.12 python3.12-venv python3.12-pip

# Verify installation
python3.12 --version
```

CentOS/RHEL/Fedora:

```bash
# For Fedora
sudo dnf install python3.12 python3.12-pip

# For CentOS/RHEL (may need EPEL repository)
sudo yum install python3.12 python3.12-pip

# Verify installation
python3.12 --version
```

Arch Linux:

```bash
sudo pacman -S python python-pip
python --version
```

Step 2: Extract Project Files

All Operating Systems

1. **Locate your project zip file**
2. **Extract to a suitable location:**

macOS/Linux:

```bash
```

```
# Navigate to desired directory (e.g., home directory)
cd ~

# Extract zip file (replace 'project.zip' with actual filename)
unzip project.zip

# Navigate into extracted directory
cd content-curator-system  # Replace with actual directory name
```

Windows:

```cmd
# Navigate to desired directory (e.g., Documents)
cd %USERPROFILE%\Documents

# Extract using built-in tools or command line
powershell Expand-Archive project.zip -DestinationPath .

# Navigate into extracted directory
cd content-curator-system
```

3. **Verify project structure:** Your project directory should contain files like:

- `app.py`
- `content_curator.py`
- `requirements.txt`
- `customers.csv`
- Various Python modules and folders

Step 3: Virtual Environment Setup

Why use a virtual environment? Virtual environments isolate project dependencies and prevent conflicts with system Python packages.

All Operating Systems

macOS/Linux:

```bash
```

```
# Ensure you're in the project directory
pwd  # Should show path to your project

# Create virtual environment
python3.12 -m venv venv

# Activate virtual environment
source venv/bin/activate

# Verify activation (should show "(venv)" in prompt)
which python
```

Windows:

```cmd
# Ensure you're in the project directory
cd

# Create virtual environment
python -m venv venv

# Activate virtual environment
venv\Scripts\activate

# Verify activation (should show "(venv)" in prompt)
where python
```

Note: Once activated, your command prompt should show `(venv)` indicating the virtual environment is active. You'll need to activate this environment every time you work with the project.

Step 4: Python Dependencies Installation

All Operating Systems

With your virtual environment activated:

```bash
```

```
# Upgrade pip to latest version
pip install --upgrade pip

# Install all project dependencies
pip install -r requirements.txt
```

If requirements.txt is missing or incomplete, install core dependencies manually:

```bash
pip install flask
pip install requests
pip install beautifulsoup4
pip install feedparser
pip install ollama
pip install sqlite3  # Usually included with Python
pip install arxiv
pip install pandas
pip install pdfplumber
pip install PyPDF2
pip install edge-tts
pip install pydub
```

Verify key installations:

```bash
python -c "import flask; print('Flask:', flask.__version__)"
python -c "import ollama; print('Ollama: OK')"
python -c "import sqlite3; print('SQLite3: OK')"
```

Step 5: System Dependencies

FFmpeg Installation (Required for Audio Processing)

macOS:

```bash
```

```
# Using Homebrew (install Homebrew first if needed)
brew install ffmpeg

# Verify installation
ffmpeg -version
```

Windows:

1. **Download FFmpeg:**
   - Go to https://ffmpeg.org/download.html
   - Download Windows build (choose "Windows builds by BtbN")
   - Extract to `C:\ffmpeg\`
2. **Add to PATH:**
   - Open System Properties → Environment Variables
   - Edit PATH variable
   - Add `C:\ffmpeg\bin`
   - Restart command prompt
3. **Verify:**

```cmd
ffmpeg -version
```

Linux:

```bash
# Ubuntu/Debian
sudo apt install ffmpeg

# CentOS/RHEL/Fedora
sudo dnf install ffmpeg  # or sudo yum install ffmpeg

# Arch Linux
sudo pacman -S ffmpeg

# Verify installation
ffmpeg -version
```

SQLite3 Installation

macOS:

```bash
# Usually pre-installed, but can install via Homebrew if needed
brew install sqlite3
sqlite3 --version
```

Windows:

1. Download from https://sqlite.org/download.html
2. Extract `sqlite3.exe` to a folder in your PATH
3. Or use: `winget install SQLite.SQLite`

Linux:

```bash
# Ubuntu/Debian
sudo apt install sqlite3

# CentOS/RHEL/Fedora
sudo dnf install sqlite

# Verify
sqlite3 --version
```

Step 6: Ollama and Granite Model Setup

Ollama Installation

macOS:

```bash
# Download and install from ollama.ai or use Homebrew
brew install ollama

# Or download installer from https://ollama.ai/download
```

Windows:

1. Download installer from https://ollama.ai/download
2. Run the installer as Administrator
3. Follow installation wizard

Linux:

```bash
# Install using official script
curl -fsSL https://ollama.ai/install.sh | sh

# Or manual installation:
# Download from https://ollama.ai/download and follow instructions
```

Start Ollama Service

macOS/Linux:

```bash
# Start Ollama service
ollama serve

# In a new terminal, verify it's running
ollama list
```

Windows:

```cmd
# Ollama should start automatically after installation
# Verify in new command prompt
ollama list
```

Install Granite Model

All Operating Systems:

```bash
```

```
# Download and install Granite 3.2:8b model (this may take several minutes)
ollama pull granite3.2:8b

# Verify model installation
ollama list

# Test the model
ollama run granite3.2:8b "Hello, please respond with 'Granite model working correctly.'"
```

Note: The Granite model download is approximately 4.7GB and may take 10-30 minutes depending on your internet connection.

Step 7: D3 Libraries for Word Clouds

The D3 libraries are typically served via CDN in the web interface, but you may need to set up local copies:

All Operating Systems

Option A: Use CDN (Recommended) The application is configured to use CDN links for D3. No additional installation needed.

Option B: Local Installation (if needed)

```bash
# Create static directory structure
mkdir -p static/js
mkdir -p static/css

# Download D3 libraries (optional - only if CDN access is restricted)
cd static/js
curl -o d3.v7.min.js https://d3js.org/d3.v7.min.js
curl -o d3-cloud.min.js https://cdn.jsdelivr.net/gh/jasondavies/d3-cloud/build/d3.layout.cloud.js
```

Step 8: Database Initialization

All Operating Systems

The SQLite database will be automatically created when you first run the application, but you can initialize it manually:

```
bash

# Ensure you're in the project directory with virtual environment activated
python -c "
import sqlite3
import os

# Create data directory
os.makedirs('data', exist_ok=True)

# Initialize database
conn = sqlite3.connect('data/content_curator.db')
cursor = conn.cursor()

# Create basic tables (the app will create others as needed)
cursor.execute('''
    CREATE TABLE IF NOT EXISTS content (
        id INTEGER PRIMARY KEY AUTOINCREMENT,
        title TEXT,
        content TEXT,
        url TEXT UNIQUE,
        source TEXT,
        timestamp DATETIME,
        relevance_score REAL
    )
''')

conn.commit()
conn.close()
print('Database initialized successfully')
"
```

Step 9: Configuration Setup

All Operating Systems

1. **Create required directories:**

```
bash
```

```
mkdir -p data
mkdir -p cache
mkdir -p podcasts
mkdir -p currentevents
mkdir -p static
mkdir -p templates
```

2. **Set up environment variables (optional but recommended):**

macOS/Linux:

```bash
# Create .env file (optional)
cat > .env << EOF
SECRET_KEY=your-secret-key-here-change-this
DATA_DIR=data
PCAST_DIR=podcasts
TOPICS_DIR=currentevents
EOF
```

Windows:

```cmd
# Create .env file
echo SECRET_KEY=your-secret-key-here-change-this > .env
echo DATA_DIR=data >> .env
echo PCAST_DIR=podcasts >> .env
echo TOPICS_DIR=currentevents >> .env
```

3. **Verify project structure:** Your directory should now contain:

```
content-curator-system/
├── venv/              # Virtual environment
├── data/              # Database and data files
├── cache/             # Customer cache files
├── podcasts/          # Generated podcasts
├── currentevents/     # Analysis output
├── static/            # Web assets
├── templates/         # Flask templates
├── app.py             # Main application
├── content_curator.py # Core functionality
├── customers.csv      # Customer data
├── requirements.txt   # Python dependencies
└── .env               # Environment variables (optional)
```

Step 10: System Verification

All Operating Systems

1. **Verify all dependencies:**

```bash
# Activate virtual environment (if not already active)
# macOS/Linux: source venv/bin/activate
# Windows: venv\Scripts\activate

# Test Python imports
python -c "
import flask
import ollama
import sqlite3
import feedparser
import requests
import pandas
print('✅ All Python dependencies working')
"
```

2. **Test Ollama connection:**

```bash
```

```
python -c "
import ollama
try:
    response = ollama.chat(model='granite3.2:8b', messages=[{'role': 'user', 'content': 'test'}])
    print('✅ Ollama and Granite model working')
except Exception as e:
    print('❌ Ollama error:', e)
"
```

3. **Test FFmpeg:**

```bash
ffmpeg -version | head -1
```

4. **Start the application:**

```bash
python app.py
```

5. **Test the web interface:**
   - Open browser to `http://localhost:5000`
   - You should see the Content Curator interface
   - Try accessing different sections to verify functionality
6. **Stop the application:**
   - Press `Ctrl+C` in the terminal where app.py is running

Troubleshooting

Common Issues and Solutions

Issue: "Python not found" or "Command not found"

- **Solution:** Ensure Python 3.12 is properly installed and added to PATH
- **Windows:** Reinstall Python with "Add to PATH" option checked
- **macOS/Linux:** Use full path: `/usr/local/bin/python3.12`

Issue: "pip not found"

- **Solution:**

  ```bash
  python -m ensurepip --upgrade
  python -m pip install --upgrade pip
  ```

Issue: "Virtual environment activation fails"

- **Solution:**
  - Ensure you're in the correct directory
  - Try creating a new virtual environment: `rm -rf venv && python3.12 -m venv venv`

Issue: "Ollama connection failed"

- **Solution:**
  - Ensure Ollama service is running: `ollama serve`
  - Check if model is installed: `ollama list`
  - Verify port 11434 is not blocked

Issue: "FFmpeg not found"

- **Solution:**
  - Verify installation: `ffmpeg -version`
  - **Windows:** Check PATH environment variable includes FFmpeg bin directory
  - **macOS:** Try reinstalling with Homebrew: `brew reinstall ffmpeg`

Issue: "Permission denied" errors

- **Solution:**
  - **Linux/macOS:** Use sudo for system installations, but NOT for virtual environment operations
  - **Windows:** Run command prompt as Administrator for system installations

Issue: "Port 5000 already in use"

- **Solution:**
  - Kill existing Flask processes
  - Or modify `app.py` to use different port: `app.run(port=5001)`

Issue: Database errors

- **Solution:**

    - Check write permissions in data directory
    - Delete and recreate database: `rm data/content_curator.db`
    - Restart application to regenerate

Issue: "Module not found" errors after installation

- **Solution:**
    - Ensure virtual environment is activated
    - Reinstall requirements: `pip install -r requirements.txt --force-reinstall`

Performance Issues

Slow Ollama responses:

- Ensure sufficient RAM (8GB minimum, 16GB recommended)
- Close unnecessary applications
- Consider using smaller model if needed

Web interface slow:

- Check database size and optimize if needed
- Ensure SQLite database is not corrupted
- Restart application periodically

Support Information

Getting Help

1. **Check log files:**
    - Application logs appear in terminal where you ran `python app.py`
    - Check for specific error messages
2. **Verify system requirements:**
    - Ensure all dependencies are properly installed
    - Check version compatibility
3. **Component-specific help:**
    - **Ollama:** https://ollama.ai/docs
    - **Flask:** https://flask.palletsprojects.com/
    - **FFmpeg:** https://ffmpeg.org/documentation.html

Maintenance

Regular maintenance tasks:

- Update pip packages: `pip install --upgrade -r requirements.txt`
- Update Ollama: `ollama pull granite3.2:8b` (to get latest version)
- Clean up log files and temporary directories
- Backup database: `cp data/content_curator.db data/content_curator_backup.db`

Next Steps

Once installation is complete:

1. **Configure RSS feeds** in the web interface
2. **Set up customer data** by updating customers.csv
3. **Run initial content curation** to populate the database
4. **Test podcast generation** features
5. **Explore chat and analysis features**

Installation Complete!

Your Content Curator System should now be fully functional. Access the web interface at `http://localhost:5000` and begin exploring the features.

For operational guidance and feature usage, refer to the user documentation provided with your project files.