Administrator Manual

MusicBug

Client
Christopher Nguyen

Team 4
Tommy Brickwedde
Jeff Clouse
Michael Cohen
Joe DiMarino
Sean Ehrig
Kevin Wiggins

4/10/2012

Table of Contents

1. Introduction

1.1 Purpose of This Document

This document is geared towards a system administrator and is a step-by-step guide to setup and install MusicBug on a linux based server. Setup is a breeze and once complete users can become fully integrated into the system. Please have the Systems Requirements Document and Systems Design Document close by for quick reference.

1.2 References

1. MusicBug System Requirements Specification
2. MusicBug Systems Design Document
3. Python (2012, April 25). Download Python. Retrieved from “http://python.org/download/”
4. Heroku Toolbelt (2012, April 25). Heroku Toolbelt. Retrieved from “https://toolbelt.heroku.com/”
5. Django (2012, April 25). How to Get Django. Retrieved from “https://www.djangoproject.com/download/”
6. SQLite Download (2011). SQLite Download Page. Retrieved from “http://www.sqlite.org/download.html”
7. pip 1.1 : Python Package Index (2012). pip 1.1. Retrieved from “http://pypi.python.org/pypi/pip”
8. Heroku (2012, April 7). Getting Started with Django on Heroku/Cedar. “https://devcenter.heroku.com/articles/django”
9. virtualenv (2012). virtualenv 1.7.1.2. Retrieved from “http://pypi.python.org/pypi/virtualenv”
10. Heroku (2012, April 16). Heroku PG Backups. “https://devcenter.heroku.com/articles/pgbackups”
11. South 0.7.4 : Python Package Index (2012) South 0.7.4. Retrieved from “http://pypi.python.org/pypi/South”

2. System Overview

The section describes the history and hardware and software requirements of the MusicBug Application.

2.1 Background

The MusicBug application is charged with creating a social music web application, which can be thought of as a mixture of iTunes and Facebook. Please refer to the Systems Requirements Specification, section 1.2, for more information about the purpose of this application.

The MusicBug application is largely self-sustaining but can be easily extended if necessary. In the event of any crash or error recovery this document describes how to quickly find log files

and generate scheduled database backups.

2.2 Hardware and Software Requirements

The system can be run on either a Windows, Mac, Linux or Unix based operating system. We support SQLite3 as the database engine for rapid software installation, with little to no additional configuration in the development environment. Python 2.6 or higher is required for proper Django operations. We recommend a 64-bit operating system with 24mb ram to process the large artist meta library and social interactions that may accrue from Facebook integration.

We recommend the use of the Heroku platform, instead of a locally supported server, to quickly launch the application with no additional changes to the settings required. Heroku does much of the heavy lifting, pushing your application to a highly scalable Postgresql database and maintaining the application on their cloud environment.

3. Administrative Procedures

The section describes how to install MusicBug software requirements as well as MusicBug itself. In addition routine and periodic administrative tasks are described below. We recommend the use of Heroku for quick deployment.

3.1 Installation

If installing on a local development or local production server there are a few prerequisites before installing and running the MusicBug web application. If running on the Heroku platform, only the installation of Heroku toolbelt and Git are required.

3.1.1 Python Installation (Local Server)

Python is required to be installed and properly configured on the system before MusicBug installation. Please refer to the “Download Python” installation procedure at “http://python.org/download/” for installing Python version 2.7 or above (Python 2012).

3.1.2 Python Package Installer Installation (PIP)

The Python Package installer provides an easy way to install packages and is recommended to quickly install Python applications as needed. Installation procedures can be found at “http://pypi.python.org/pypi/pip” (pip 2012).

3.1.3 virtualenv (Python, PIP dependent)

virtualenv allows the creation of isolated Python application environments (virtualenv 2012). If pushing to Heroku, virtualenv is required for Python based Heroku applications. Please refer to “http://pypi.python.org/pypi/virtualenv” for installation instructions with pip.

3.1.4 Django Installation (Python dependent)

MusicBug takes advantage of the python based Django content management system (CMS).

Please refer to the Django installation instructions at “https://www.djangoproject.com/download/” to quickly install Django (Django 2012).

3.1.5 Heroku Toolbelt (Includes GIT, Heroku Client, and Foreman)

Running on the Heroku platform requires pushing the local MusicBug application to a Heroku server. The heroku toolbelt includes Heroku’s custom client, for heroku command line tasks, Foreman, for local development and testing, and GIT, for easy distributed version control and deployment via git command line or GUI operations. Please refer to Heroku’s documentation at “https://toolbelt.heroku.com/” to quickly install and configure all three tools (Heroku Toolbelt 2012).

3.1.6 SQLite3 (Required for local servers)

If running a local test copy we suggest the use of SQLite3 for easy setup. Please follow SQLite instructions at “http://www.sqlite.org/download.html” to quickly install SQLite3 on the appropriate system.

3.1.7 South (PIP and Python Dependent)

South is a database migration tool for Python. Please follow the south installation with pip tutorial at “http://pypi.python.org/pypi/South” to install South on your system.

3.1.8 MusicBug

Once all prerequisites are installed, MusicBug installation is fast and easy.

3.1.8.1 Installing for Heroku Deployment

To install MusicBug copy the heroku_v directory contents into a Heroku application folder initialized with Heroku create on the Cedar stack. This folder will serve as the root with GIT distributed version control. The copied files include a previously created isolated Python environment with a requirements file containing the python applications for the Heroku environment. Use the git push command, “git push heroku”, to push the application to Heroku.

3.1.8.2 Installing to Local Server

When setting up a framework on a local server, copy the music folder onto the server and run the music application with python. However, we highly recommend the use of heroku.

3.2 Routine Tasks

Moderating reviews must be completed by an administrator. As reviews are submitted, an administrator must approve or delete each review. Until a review is approved, the review is only viewable for the user who wrote the review. For additional security Administrators can only be set manually by updating the User table, setting moderator to True for that specific user. Moderator privileges include approving or deleting reviews to prevent spamming and trolling.

Depending upon the environment additional tasks may be required.

3.3 Periodic Administration

We recommend periodically backing up the system in the case of server failure and to prevent loss of user information. We recommend using Heroku Postgresql backup tool to setup automatic database backups (Heroku April 2012) if using Heroku. Please refer to Heroku documentation at "https://devcenter.heroku.com/articles/pgbackups".

3.4 User Support

For additional support please contact us via email. We are always available with 24 hour customer service representatives standing by.

4. Troubleshooting

In the event of an error or bug troubleshooting is necessary. We have outlined common issues that may arise during use of the MusicBug application.

4.1 Dealing with Error Messages and Failures

Please contact our online support forums in troubleshooting all error messages or failures. A common issue that often arises is the need for additional database columns or tables. In the event of changes please ensure appropriate migrations and syncing occurs as required by python.

4.2 Known Bugs and Limitations

Currently only a few bugs exists in the system. On login with facebook, intermittently facebook returns an undefined value to our local database system. However, the local system verifies the message is from facebook and is legitimate using the appropriate secure key authentication but the login fails to provide required user meta data. When this occurs clearing the browser session and/or deleting all cookies and history will allow the user to log-in properly. Another limitation is with the 7digital API call to determine if a song is explicit. Currently their API does not always find the explicit song. In this scenario there is the possibility for a track to be explicit but not show up as explicit.

4. Appendix A – Team Review Sign-off

This document has been collaboratively written by all members the team. Additionally, all team members have reviewed this document and agree on both the content and the format. Any disagreements or concerns are addressed in team comments below.

Team

Name _____ Date _____
Print
Name _____ Date _____
Signature
Comments _____
_____

Name _____ Date _____
Print
Name _____ Date _____
Signature
Comments _____
_____

Name _____ Date _____
Print
Name _____ Date _____
Signature
Comments _____
_____

Name _____ Date _____
Print
Name _____ Date _____
Signature
Comments _____
_____

Name _____ Date _____
Print
Name _____ Date _____
Signature
Comments _____
_____

Name _____ Date _____

Print
Name _____ Date _____
Signature
Comments _____
_____

5. Appendix B – Document Contributions

Michael Cohen documented and completed the administrator manual for MusicBug. He is the key owner of this document completing 70%. Jeff Clouse edited the document and added additional content completing 20%. Sean Ehrig edited the document and added additional content completing 10%.