Installation and Maintenance Guide

Library Website

Contents

[Getting Started 2](#_Toc121468754)

[Local Requirements 2](#_Toc121468755)

[Heroku Requirements 2](#_Toc121468756)

[Installation 3](#_Toc121468757)

[Deployment 7](#_Toc121468758)

[Local Deployment 7](#_Toc121468759)

[Heroku Deployment 8](#_Toc121468760)

[Maintenance 11](#_Toc121468761)

[Website Publishing 11](#_Toc121468762)

[Quality Assurance 11](#_Toc121468763)

[Contact Monitoring 12](#_Toc121468764)

[Infrastructure Performance Monitoring 12](#_Toc121468765)

[Development 13](#_Toc121468766)

[Models 13](#_Toc121468767)

[CSS 14](#_Toc121468768)

[Glossary 15](#_Toc121468769)

# Getting Started

Before installing the website on your local machine, make sure you have all of the requirements, listed below.

## Local Requirements

* Python 3

To install the website on your local machine, you need to have Python installed.

For instructions install Python, follow the official guide at the following link: <https://wiki.python.org/moin/BeginnersGuide/Download>

## Heroku Requirements

As well as the local requirements, running the website hosted with Heroku has all of the requirements listed below. *If you don’t plan on running the website via Heroku, you can skip these requirements.*

* Heroku Account

To host your website on Heroku, you need to first create a Heroku account.

You can sign up with Heroku at the following link: <https://signup.heroku.com/>

Once you have an account on Heroku and have logged in, click the “Create new app” button. Give your app an appropriate name, as it will be the app name for your website, then click “Create app.”

* Heroku CLI and Git

The Heroku CLI allows you to access your website when it is deployed on Heroku. Using the Heroku CLI also requires Git to be installed on your local machine.

For instructions to install the Heroku CLI, follow the official guide at the following link: <https://devcenter.heroku.com/articles/heroku-cli#install-the-heroku-cli>

For instructions to install Git, follow the official guide at the following link: <https://git-scm.com/book/en/v2/Getting-Started-Installing-Git>

* (Optional) GitHub and GitHub Account

Heroku allows you to deploy your website manually or automatically from a repository on GitHub.

For instructions on how to use Heroku with GitHub, follow the official guide at the following link: <https://devcenter.heroku.com/articles/github-integration#manual-deploys>

You can sign up with GitHub at the following link: <https://github.com/signup>

# Installation

Following is a detailed list of instructions for installing the library website:

1. Install pip

Pip is a package manager to manage your Python code libraries. To install pip, open your command prompt and type one of the following:

|  |
| --- |
| Windows: |
| py -m pip install --upgrade pip |
|  |
| Unix/macOS: |
| python -m pip install --user --upgrade pip |

If you have trouble installing pip, follow the official alternatives guide at the following link: <https://pip.pypa.io/en/stable/installation/>

1. Install virtualenv

To keep all of your packages in an isolated location, you need to set up a virtual environment for Python. One of the most common virtual environments for Python is called virtualenv, although you can use an alternative that works with Python. To install virtualenv, open your command prompt and type one of the following:

|  |
| --- |
| pip install virtualenv |

1. Create a Virtual Environment

Once you have install virtualenv or the virtual environment of your choice, you need to create a new virtual environment. The following instructions will be for using virtualenv. Type the following into your command prompt:

|  |
| --- |
| virtualenv <VIRTUALENV DIR> |
|  |
| <VIRTUALENV DIR> being replaced with the file path to your new virtual environment, which you should put in the website folder. |

1. Installing Website Required Packages

To install the remaining required packages, you need to activate your virtual environment. The virtual environment directory should be located within your website folder. Change the directory of your command prompt to the location of your website folder with the following command:

|  |
| --- |
| cd <WEBSITE DIRECTORY> |
|  |
| <WEBSITE DIRECTORY> being replaced with the file path to the website folder. |

Next, activate your virtual environment with the following command:

|  |
| --- |
| Windows: |
| <VIRTUALENV DIRECTORY>\Scripts\activate |
|  |
| Unix/macOS: |
| source <VIRTUALENV DIRECTORY>/bin/activate |

Once your virtual environment is activated, you can install the website required packages using pip. Type the following command in your terminal:

|  |
| --- |
| pip install -r requirements.txt |

1. Install PostgreSQL 14

To run your website on your local machine with the provided database, you need to install PostgreSQL 14 into your virtual environment. To install PostgreSQL 14, go to the following link: <https://www.postgresql.org/download/>

Select the correct installer for the operating system you are using. Once it is finished downloading, open the executable file and start the setup.

***! IMPORTANT*** – When setting up, PostgreSQL will ask you to set up a password for your databases. Be sure to keep this password in a safe place as you will need it in future steps.

1. Create a Local Database

To run the website locally, you need to create a new PostgreSQL database to hold all of the information in the SQL file in your website folder. Before being able to access the database, you need to create a new one. To do so, type the following command:

|  |
| --- |
| create <DATABASE NAME> |
|  |
| <DATABASE NAME> is replaced with the name you want your database to have. |

When creating a new database, it will ask for a password. This is the password you set when you installed PostgreSQL in the previous step.

Once your new database is set up, you can now import the information for the library website into it. To do so, type the following command:

|  |  |
| --- | --- |
| psql -U postgres -h localhost -d <DATABASE NAME> -f db.sql | |
|  | |
| -U postgres signifies your username. Unless you changed it, the default is “postgres.” | |
|  | |
| -h localhost is the location on which you are hosting your database. Unless you changed it, the default is “localhost.” | |
|  | |
| <DATABASE NAME> is the name of the database you created in the previous step. | |
|  | |
| -f db.sql is the file that contains all of the database information for the library website. | |

After running this command, it will prompt for a password. This is the same password you set up when you installed PostgreSQL.

1. Updating Website Settings with Database Information

Before you can deploy your website locally, you need to update the database information in the library website settings.

Open the folder containing the website files. Find the folder called “locallibrary” and open the settings.py in a text editor. Find the section called #Database, which should look like the following:

|  |
| --- |
| DATABASES = {  ‘default’: {  ‘ENGINE’: ‘django.db.backends.postgresql\_psycopg2’,  ‘NAME’: ‘<database name>’,  ‘USER’: ‘<username>’,  ‘PASSWORD’: ‘<password>’,  ‘HOST’: ‘127.0.0.1’,  ‘PORT’: ‘5432’,  }  } |

You need to update parts of this section to work with the database you set up in the previous steps:

<database name> should be the name of the database you created in the previous steps.

<username> should be the username you set up when you installed PostgreSQL. If you did not change the username, the default is “postgres.”

<password> should be the password you chose when you first installed PostgreSQL.

Once these steps are successfully completed, you should be able to run your website on your local machine. To see how to run your server on your local machine, see the *Local Deployment* section under *DEPLOYMENT*.

# Deployment

## Local Deployment

Once you have finished installing and setting up your website in the previous sections, you can now deploy your website on your local machine. Following is a detailed list of instructions for deploying the library website locally:

1. Run Server

To run your server locally, open a command prompt. Change the directory to the website folder and activate your virtual environment (for more information on running your virtual environment, see *INSTALLATION* step 4).

Now, you can start running your server. To start the server, type the following command:

|  |
| --- |
| python manage.py runserver |

This command starts the server running on your local machine.

1. View Your Running Website

Once you have started your server, you can view it on your local machine. Open a web browser and type the following into the URL space:

|  |
| --- |
| 127.0.0.1:8000 |

This will pull up the website in your browser, which is now running locally. You can interact with the website as you would if it were hosted on the internet.

## Heroku Deployment

All of the website files have already been adjusted to work with Heroku. You can deploy with Heroku either with Git or GitHub. Following is a detailed list of instructions for deploying the library website on Heroku:

1. Make a Git Repository

To make the website directory into a repository, you need to use Git to initialize it. Open a command prompt, change the directory to website folder, and type the following command (for more information on running your virtual environment, see *INSTALLATION* step 4):

|  |
| --- |
| git init |

This creates a repository that we can use with Heroku.

1. Make a Git Ignore File

In order not to commit your virtual environment to Heroku, you can make a git ignore file in your repository. To do so, create a text file named “.gitignore”.

After creating the file, open it in a text editor and add the following:

|  |
| --- |
| <VIRTUALENV DIRECTORY>/ |
|  |
| <VIRTUALENV DIRECTORY> being replaced with then name you gave to your virtual environment folder. |

1. Add Your Files to Git

To be able to push your repository to Heroku, you first need to commit your files to the local repository you created in step a. To do so, type the following commands:

|  |
| --- |
| git add . |
|  |
| git commit -m “Initial commit” |

These commands add and then commit your files to your local repository. Now, you’re ready to work with Heroku.

Once you’ve added your website files to a Git repository, you can either deploy on Heroku via Git or GitHub. Following is a detailed list of instructions for deploying the library website via Git and GitHub:

1. Git
   1. Login to Heroku CLI

Before you can push your repository to Heroku, you have to login to the Heroku CLI. To do so, type the following command:

|  |
| --- |
| heroku login |

The CLI will ask you to press a button to login and will open a browser for you to enter your credentials. Once logged in to Heroku, you can proceed.

* 1. Push to Heroku

Finally, you can push your website repository to Heroku and update your Heroku database. Make sure the db.sql database file is located in the root folder of your website directory. To do so, use the following commands:

|  |
| --- |
| heroku git:remote -a <APP NAME> |
|  |
| git push heroku master |
|  |
| heroku pg:psql -a <APP NAME> < db.sql |
|  |
| <APP NAME> being replaced by the name you gave to your app on Heroku. |

Once you’ve pushed your files to Heroku and updated the database, you can press the “Open app” button on the Heroku website of your app and view the website live.

1. GitHub
   1. Create a GitHub Repository

Logged in to your GitHub account, create a new repository. Give it a meaningful name as it will be the repository for your website. Click “Create Repo” at the bottom.

You can add your files to this repo either through the use of Git or by manually uploading them.

* + 1. Git

Make sure your command prompt is in your website folder. Use the following commands to add your files to GitHub:

|  |
| --- |
| git remote add origin <REPO URL>.git |
|  |
| git branch -M master |
|  |
| git push -u origin master |
|  |
| <REPO URL>.git replaced by the URL for your repository.  Ex: https://www.github.com/YourUsername/RepoName.git |

* + 1. Manual Upload

On your repository’s GitHub page, there is a link called “uploading an existing file” under the “Quick setup” section. Select the link and you can either drag your files into the browser to upload them or click the “choose your files” button and you can select the folder from your computer.

* 1. Connect GitHub to Heroku

On the Heroku website, logged into your account, and on the app you created for your website, click the “Deploy” button at the top. Further down the Deploy page, you’ll find a section called “Deployment method.” Here, you can add your GitHub account to Heroku, which will require you to give Heroku permission to access your GitHub repository.

* 1. Deploy

Once your Heroku app has been linked to your GitHub repo, you can deploy your website on Heroku. On the same page that you connected GitHub to your Heroku app, there are two sections called “Automatic deploys” and “Manual deploy.”

* + If you want your website to be automatically updated every time you change files in your GitHub repository, you can select the master branch under “Choose a branch to deploy” and then click the “Enable Automatic Deploys” button.
  + If you want to manually deploy your files from GitHub whenever they are changed, you can do so under “Manual deploy.” Select the master branch under “Choose a branch to deploy” and click the “Deploy Branch” button.

Once you have done either of these options, under the “Activity” tab of your Heroku app, you can see the progress of your deployment. To see the newly deployed website, press the “Open app” button at the top of the screen.

# Maintenance

## Website Publishing

Keeping your website running and updated is an important part of website maintenance. The responsibilities of maintaining a published website are covered by several different categories.

* Editing/Adding Content

It is important to keep everything on the website up-to-date and accurate. Any time there is a change to the catalog, library contact information, or library hours, the website must be edited to display the most recent information.

* Design

The most significant aspect of a website is its visual and interactive presentation.

This doesn’t only apply to aesthetics, but also to accessibility. As new rules and regulations come out about the accessibility of websites, the website needs to be kept updated with these new requirements.

* Developing

Building the content and design into HTML files is the responsibility of a developer. Adding new features, changing current features, creating more interactivity, etc. all fall under this category.

## Quality Assurance

Quality assurance is the process of keeping the website free of errors, mistakes, or bugs.

The process of collecting data for quality assurance purposes consists of examining the website for any issues. This includes checking for the following:

* + Broken links
  + Missing content and images
  + Spelling and grammar
  + Browser compatibility
  + Application functionality
    - Account login
    - Account signup
    - Contact form
    - Book comments
    - Book renewals
  + Accessibility conformation
  + Standard design

Issues found while collecting quality assurance data should be sent to the developer for correction or update.

## Contact Monitoring

Managing correspondence received from the Contact page of website is a vital part of website upkeep. It is important to acknowledge and respond to client messages in a timely manner.

Contact can also come in the form of comments on any of the books in the library catalog. Thus, going through the comments and checking the messages’ relevance and appropriateness.

However, responding to messages is not just about answering questions about the library. There may be some issues that arise from the website that are not in quality assurance testing. The following steps were put in place to help you manage the website’s contact messages:

1. Collect the information
2. Acknowledge receipt
3. Investigate
4. Respond and follow-up

## Infrastructure Performance Monitoring

While the library website was designed to either be hosted on a local machine or via Heroku, it is still important to keep the systems running the website up-to-date and free of oversights. This includes monitoring the health of the following systems:

* Server response times
* End-User response times
* Server error rates and types
* Traffic patterns
* Timeouts

If the website is deployed via Heroku, most of the above are managed by Heroku. However, it is still important to keep on top of these issues and have them resolved with Heroku’s staff should some issues arise.

# Development

Most of the development of the website is as simple as changing the text in files or managing catalog content via the library admin website. However, there are a few instances of updating the library that require a different approach.

## Models

The models (located in models.py in several of the subfolders of the website directory) are a collection of definitions for storing data. For example, the author model has defined information variables such as name, date of birth, etc. The models tell the database what information is stored in each table. In the previous example, author would be a table with rows for name, date of birth, etc.

Should you want to change the models to add more information, change how information is managed, or completely adding a new model, there is a specific process you need to go through to update your database to reflect these changes. Following is a detailed list of instructions for update the models and the database:

1. After making the changes to the models.py file, you need to make the “migrations” to the database. Open your command prompt and change the directory to your website folder then activate your virtual environment (see Installation, step 4 for how to change directory and activate your virtual environment).
2. There are two steps to making the migrations to the database file on your local machine (db.sql). The first is called makemigrations, which packages up all of your individual changes into a file that can be easily read by your database. The second is called migrate, which does the actual job of adding your changes to your database. To activate these commands, type the following in the command prompt:

|  |
| --- |
| python manage.py makemigrations |
|  |
| python manage.py migrate |

Once these changes are finished being made, you can run your server and see how the changes have taken effect.

1. (Optional/Heroku) If using Heroku, you need to apply these changes to your Heroku database as well. This can be done via the Heroku CLI. After following the previous two steps to make the migrations on your local machine, you then make the changes on your Heroku app. Log in to the Heroku CLI, push your code to Heroku (see Heroku Deployment for information), then type the following command:

|  |
| --- |
| heroku run -a <YOUR APP NAME> python manage.py migrate |
|  |
| <YOUR APP NAME> being replace with the name you gave your app on Heroku. |

This will migrate the changes on your Heroku database.

## CSS

The library website uses a framework to help efficiently write the CSS (cascading style sheet) of the website called Tailwind. This makes it easier to change how each page is displayed and designed. In order for your changes to take effect, you need to have Tailwind running. Every time you save a change to an HTML file, Tailwind automatically takes your changes and adds them to the stylesheet of the website.

To start using Tailwind while editing your website, use the following steps:

1. Open your command prompt and change the directory to your website folder then activate your virtual environment (see Installation, step 4 for how to change directory and activate your virtual environment).
2. Start the Tailwind service by typing the following into your command prompt:

|  |
| --- |
| python manage.py tailwind start |

This activates the Tailwind service and you can now start changing the styles of your HTML files. Tailwind uses specific keywords kept in the class of each page element to determine what changes need to be made. See the official Tailwind documentation for more information: <https://tailwindcss.com/docs/>

Every time you save your HTML file, Tailwind updates, so be sure to save often for the changes to take effect.

1. Once finished editing the HTML pages, press Ctrl + C in your command prompt to stop the Tailwind service.

# Glossary

**command prompt** – the program in which you can input text to interact with programs on a computer. Also known as cmd, cmd.exe, command line interface, and terminal.

**database, db.sql** – the collection of tables holding the information for the content of the library.

**git** – a tool used for managing source code. Git is run directly on your computer.

**GitHub** – similar to git, a tool used for managing source code. GitHub is mainly run from the GitHub website.

**gitignore** – the file in a repository that tells git which files to ignore when pushing to a repository (ex: ignoring your virtual environment directory)

**Heroku** – a could application platform that allows clients to build, run, and operate applications in the cloud or without hardware of their own.

**Heroku CLI** – the program installed on your computer that allows you to interact with Heroku through the command prompt.

**local machine** – your computer.

**pip** – a tool for managing Python code library packages.

**PostgreSQL** – database management software.

**Python** – a programming language in which the library website was mainly built.

**model** – a defined set of how you can store information to make up a whole. For example, the book model of the library has defined information as title, author, genre, etc.

**migration** – solidifying the changes you’ve made to your application in the database. Migrations ensure that your database has the tables, columns, and rows to keep all of your information.

**repository** – a place where project files are stored. A repository tracks the history of changes made to the files within the repository folder.

**virtual environment, virtualenv** – a directory of packages enabling you to develop the website in a secluded part of your computer.