Secure and Remote 3D Printing User/Developer Manual

Tiffanie Petersen - tpetersen2018@my.fit.edu

Isaiah Thomas - ithomas2018@my.fit.edu

Carl Mann - cmann2013@my.fit.edu

Nick Contrell - ncontrell2019@my.fit.edu

Sponsored by

Dr. Siddhartha Bhattacharyya - sbhattacharyya@fit.edu

Table Of Contents

1. User Guide

1.1 Getting started

Our project was created with two specific end-users in mind; students and faculty. The first goal was to provide a functioning web application with which normal users can upload files, view their project(s) status in a queue, and log in or log out. For administrative users, we wanted to ensure the ability to display the status of printing requests, accepting or declining printing requests, and allowing for profile bans. A basic demonstration of creating an account may be found here. All accounts are required to register with a valid Florida Tech email address and are validated by administrators before printing jobs are accepted.

1.2 Accessible pages

Students:

Students are only shown pages to which they are allowed to navigate. On these pages we allow students to register, login, logout, upload files, and see their current account status. Below are some brief descriptions of pages accessible to students.

User Accessible Pages	**Description**
Home / User Acknowledgement	Main page of the website displays general information as well as directs users to other pages. The web application will display a greeting message based on the signed-in user's information.
Registration	Allows a user to create a username and password and provide a Florida Tech email. Once this information is approved they will be allowed to login and submit printing requests.
Login	Page for user authentication. Functional for all user types.
Logout	Option provided along with navbar after

	successful sign-in that allows users to log out.
Upload	Webpage that allows users to upload a GCODE file and submit it to the server as a print request.
Navbar	A navigation bar displaying links to various pages on the site. Used for traversal of the website.

Faculty:

Administrators can either have their account created as a superuser while the website is offline, or an existing administrator can use the Admin portal to create their account. Faculty have all of the access student users possess with the following additional pages:

Admin Accessible Pages	**Description**
Home	Main page of the website which displays general information as well as directs users to other pages.
Queue	The queue displays relevant data to administrators regarding printing requests. Some data on the queue page includes file upload time, the user who uploaded a specific file, and the file's name. Administrators may also start, stop, pause, and cancel print jobs from this page.
Model viewer	Each print job in the queue may be viewed on the model viewing page. Here, the models are rendered into a 3D space which may then be viewed by administrators. From this viewing, administrators may decide whether or not to

	print a requested job.
Admin panel	This page provides administrators with their own UI for viewing models within the database as well as user-created accounts. From this page users have the ability to remove users, escalate their privileges, and manage models.

1.3 User operations and features

Students:

Students are capable of navigating and interacting with the website in various ways. Several buttons which initiate some form of interaction with the server are listed below. Their operations are described in detail and the circumstances in which these buttons may be used are outlined.

Default user operations and features	**Description**
Navbar (all pages)	- Home button - Register button - Login button - File upload button
Register button (registration page)	Sends a request to the server to create an account with a given set of information and credentials. Various checks are then made by the server to guarantee: - This is not a duplicate user - The email address is valid for FIT - The password is of a high enough difficulty - None of the data in each field surpasses the length of the variables

	they will be stored in There is currently no verification process implemented for user account creation other than checks on the data provided. Due to the website application being deployed solely within a restricted WLAN, developing SMTP verification tokens was deemed unnecessary for project objectives.
Login button (registration page)	Sends a request to the server with the information provided. If the data matches that which we have on record, the user is redirected to the home page and welcomed as the respective user. If the data does not match a recorded user, they are redirected back to the login screen to try again.
File upload button (upload page)	Sends a request to the server to add a file to the queue to be printed. In order for this button to successfully send a request, the user must be logged in. Required fields for uploading include a valid title and gcode file. Files are investigated rigorously and may be rejected before or after file upload.

Faculty:

Alongside the functionality provided to basic users, faculty are trusted with built-in functionality that has the potential to impact the data stored on the server as well as the operation of the printer.

Admin operations and features	**Description**
Navigate to admin panel button (home page)	Redirects an authenticated user to the admin page of the website.
Create account button (admin panel)	Administrators may create an account with either default credentials or administrative

	credentials from the admin panel.
Delete account button (admin panel)	Administrators may delete accounts from the database via a button on the admin panel.
Edit account button (admin panel)	Allows administrators to edit the account status and information of any user in the database. Located on the admin panel's list of users.
File removal button (queue page)	Sends a request to the server to remove the file of a given name in the queue from both the queue and the database.
Printer home button (queue page)	Sends a call to OctoPrint via REST API which communicates to the printer that it must begin its homing process to the designated starting coordinates.
Printer start print button (queue page)	Sends a call to OctoPrint via REST API which communicates to the printer that it must select the print with the same name in the queue and start the print job.
Printer stop print button (queue page)	Sends a call to OctoPrint via REST API which communicates to the printer that it must stop its current print job.
View model button (queue page)	Brings an administrator to the model viewing page. Each item in the queue has a unique link for the model viewer.

1.4 Security

The scope of this project was to create remote printing capability for the printer via a web application and ensure the security of the 3D printer from cyber threats. Some key security measures implemented to protect remote users and the client in this project include HTTPS, CSRF tokens, uploaded file screening, authorized user whitelist, administrative control of prints, and non-exposed local handling of files between the

printer and the site. While our application is tailored to the specific infrastructure in the lab, we have designed a platform that may be deployed and scaled up with relative ease should there be a demand for printing services.

The purpose of this project was to provide students and faculty outside of the printer lab an ability to request their 3-D model be printed while ensuring the protection of the lab's Ender-3 printer and its network. This was accomplished by deploying a secure web application that allows authenticated users to upload GCODE files that are placed in a print queue by administrators. Throughout development, our team maintained regular communication with clients and faculty in order to keep fulfilling expectations and stay on track with their vision. When it comes to the importance of our client's work we are looking to mitigate situations in which injection or exfiltration may occur.

2. Developer Guide

2.1 Installation

Setting up SSH for GitHub:

- Git clone git@github.com:IsaiahST2020/SeniorDesignProject.git
- Navigate to ~/.ssh
- ssh-keygen -t rsa -b 4096 -C "SeniorDesign NAME"
- Then copy the generated id_rsa.pub to Github
- Finally, enter the following command in your cloned repository
    - git remote set-url origin git@github.com:IsaiahST2020/SeniorDesignProject.git

Setting up pipenv:

- To have this environment run as smoothly as possible, you can run the following commands to get a working setup for production or development. To get all the dependencies independent of the system, you will need to install pipenv

debian/ubuntu:

sudo apt install pipenv

every other system:

$ pip install pipenv

2.2 Dependencies

Installing dependencies with pipenv:

- Once that is done, navigate to the workspace and run the following commands

  $ pipenv install

Installing dependencies and development libraries:

- This includes things like pylint and autopep8 to help keep formatting and documentation consistent.

  $ pipenv install --dev

2.3 Setup

Running the application:

- If you want to run the server using the virtual environment provided by pipenv you can follow these steps on a fresh install

```
pipenv uninstall --all
pipenv install --dev
pipenv shell
sudo python3 -m pip install -r requirements.txt
pip freeze
mkdir data
cd data
touch db.sqlite3
cd ..
sudo python3 manage.py makemigrations
sudo python3 manage.py migrate
sudo python3 manage.py createsuperuser
sudo python3 manage.py runserver
```

- And for post-setup development

```
pipenv shell
exit (to leave shell)
```

2.4 Local development

The website runs on the Django framework. One great resource we found for picking up how to use this framework and its key features was the video here. This video covers how Django operates. Some of the topics covered include models, views, databases, and security measures. Most of the local development for this project was done using sublime on kali Linux virtual machines. On the Github repository, you may see both sublime and visual studio configuration files to ensure your editor uses spaces rather than tabs.

Below is a layout of the structure and codebase of the project:

File or code block	**Description**
Manage.py (root file)	This file starts the server and handles the initial connection with the OctoPrint API. By running the command “python3 manage.py help” you can see all of the possible commands supported by Django. The command to start the server with this file is “python3 manage.py runserver”. Manage.py isn’t changed often and has little effect on the overall project other than starting the process.
Admin.py	Admin.py controls the backend operations of the admin panel page of the website. This page is standard for many Django web applications and provides administrators with many great tools to moderate the site.
Settings.py	Settings.py is an incredibly important file in which definitions for paths, libraries to be imported, and routes to be accessible to users are handled.

	This file is changed relatively often as new imports are needed for functionality. One big part of settings.py is the definitions of the project's base, template, and static directories.
Views.py	Views.py is how the server handles and serves requests from the users who wish to view certain pages. Each definition in views.py corresponds to a different page of the website and different handling of data provided by the user. Views.py can handle various types of requests such as GET, POST, and HEAD. It may also redirect users to other pages once developer-defined conditions are met.
Models.py	Models.py is where all of the file and user models are defined. Each model holds a set of fields. Models are stored in the database and may be viewed on the admin panel page. Many fields inside of a model are not allowed to be left empty which is why in some cases you may see a default value or the explicit assignment of an empty value when none is provided. Once this file has been changed it is necessary to make migrations as described in the next section for pushing changes.
(base…upload).html	Stored in the /templates/ directory, all of the html pages served by the website fall under this category. Django has many useful plugins which allow it to interact with html files in unique ways and inject scripts for methods such as table creation. This idea is elaborated upon further in the video provided earlier in the manual.

The GreatFET is one of the tools we used to test the security of the project. The GreatFET acts as a proxy for USB connection and data can be handled as a developer sees fit once passed through it. Several attempts were made to MITM/fuzz the connection between OctoPrint and the printer however so far the USB to serial connection does not seem to interact well with the prebuilt modules of the GreatFET. Here and here are some good resources for learning more about the GreatFET and its supporting libraries. With some additional work we believe the printer may be fuzzed and tested even more extensively.

2.5 Pushing changes

In order for changes related to models and various other important structures in the Django framework to be realized a user must migrate them with two simple commands.

- Python3 manage.py makemigrations
- Python3 manage.py migrate

Most other changes take effect immediately upon saving. An example of changes which take effect during runtime include html, views, and admin files.

2.6 Diagrams

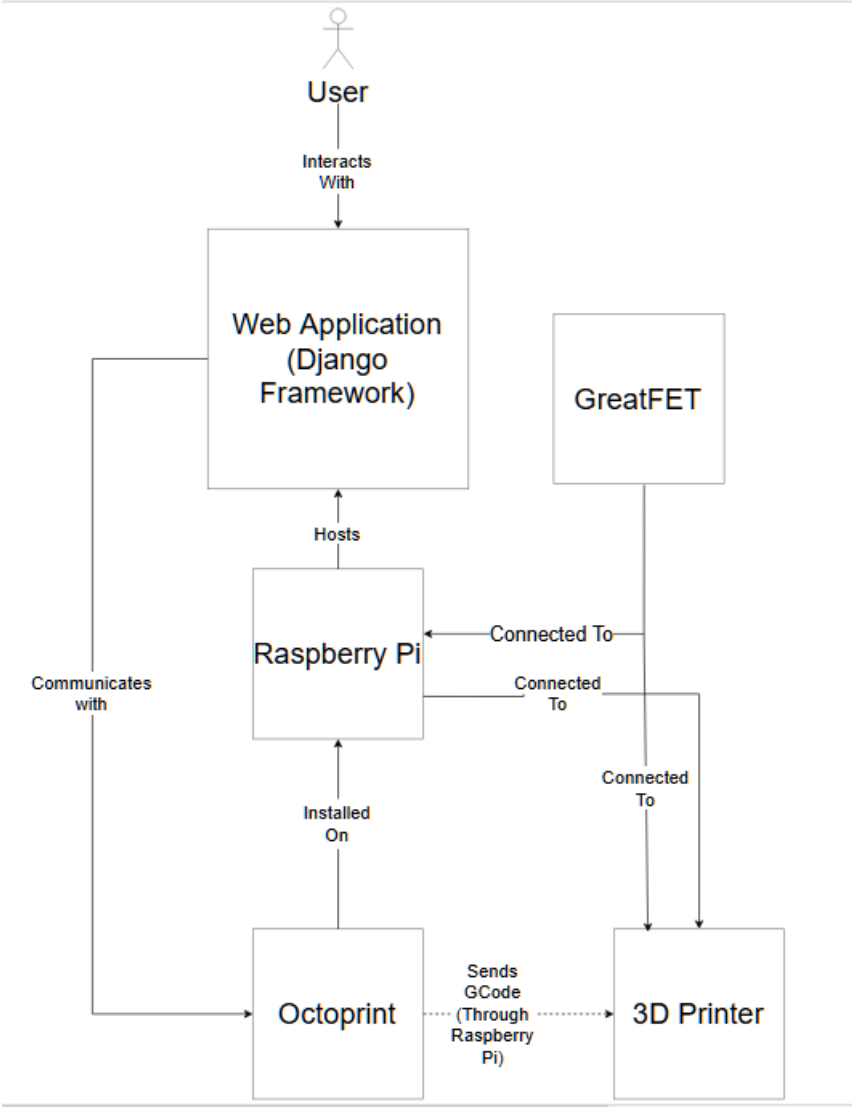

Figure 2.7.1: Framework diagram

The framework design shows how the software and hardware interact with each other. The main part of this project takes place with the Raspberry Pi because it stores all of the data from the web application, runs the docker image, interfaces with octoprint, and connects to the 3D printer and the GreatFET.

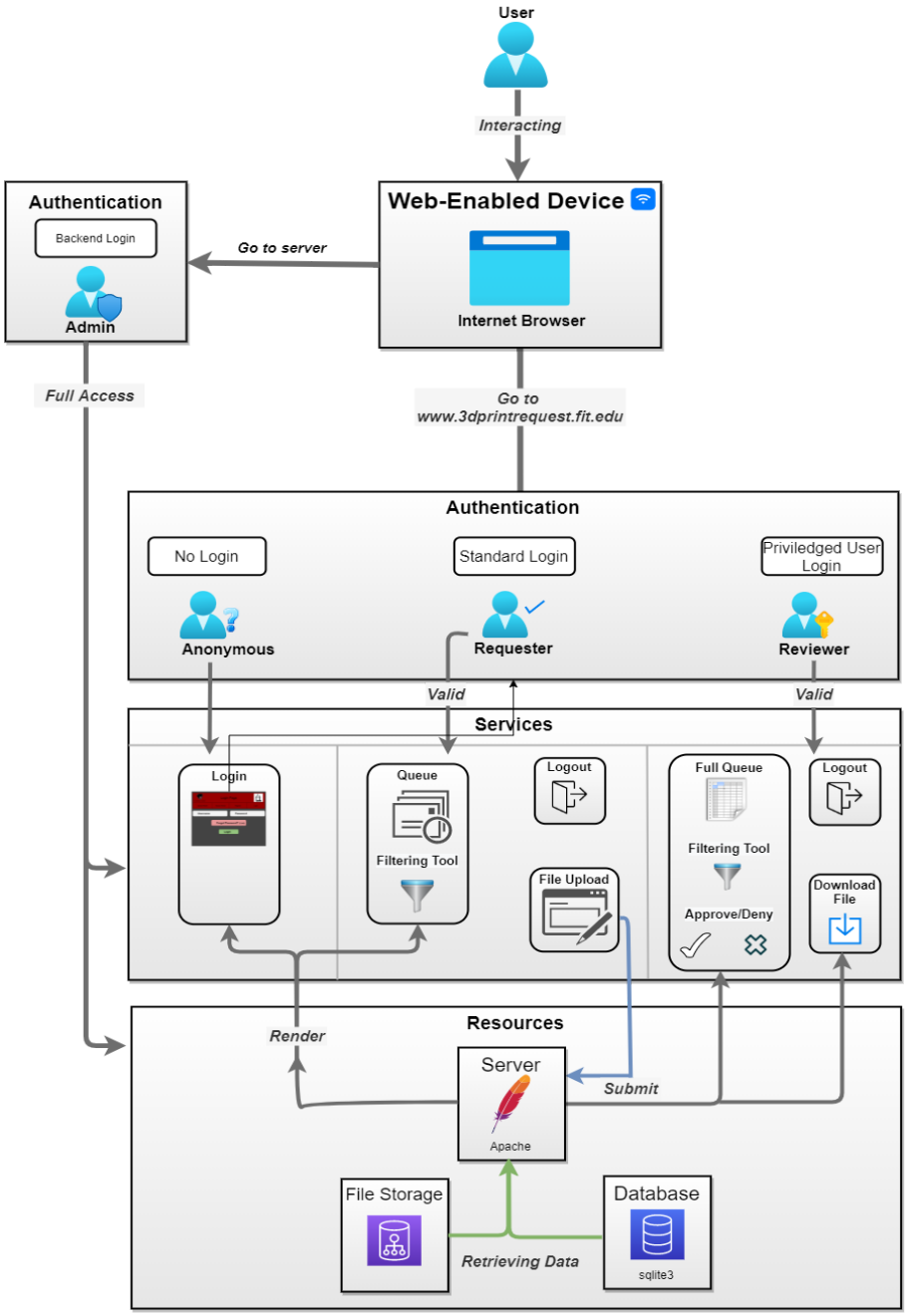

Figure 2.7.2: Web services diagram

Figure 2.7.2 shows how a user will interact with the web server. An unauthorized user will have limited access to the web application to keep the traffic to a minimum while also ensuring that sensitive information cannot be accessed by the public. The next type of user will be authorized or logged in, which will allow the user to see more webpages and gain access to uploading files which an administrator can print. The Privileged logged in user (or administrator) will be able to see a

full queue of all the projects waiting to be approved and printed as well as deleting users, downloading files from the databases and more.

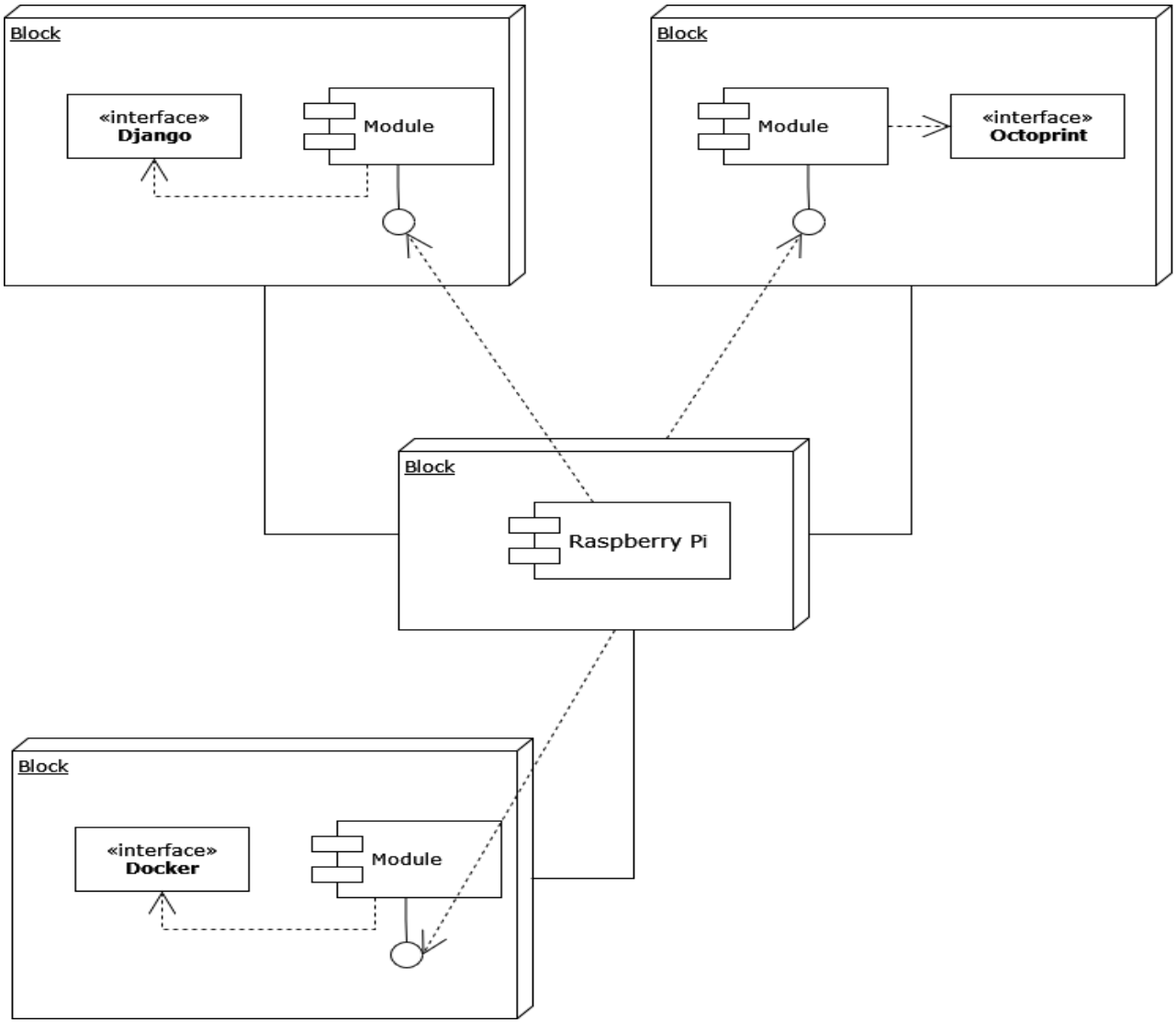

Figure 2.7.3: Software deployment diagram

The software that is needed for secure and remote 3D printing include Docker, Octoprint, and Django. These three software products will be downloaded onto a Raspberry Pi to interact with each other to provide the remote experience of 3D printing. Django will interface with Octoprint to ensure that each project fits the specifications of the 3D printer and ensures that the gcode file is not harmful to the printer.

3. Deployment Guide

3.1 Printer

An excellent guide for setting up an Ender3 may be found here. The guide covers more than enough information to use as a reference during the maintenance of the Ender3. We recommend that you find an experienced user of the Ender3 to help with calibration, troubleshooting, and testing. On the repository and the raspberry pi currently hosting the demonstration setup, there are calibration gcode files that may be used to ensure the printer is in working order.

3.2 Raspberry Pi

A raspberry pi 4 is a great choice for hosting the website and OctoPrint as it is Wifi ready after setting up Raspbian. By containerizing each of the applications on the raspberry pi we effectively create a secure environment by design. Unless an attacker has gained access to the pi itself, API communications between the website and OctoPrint are secure by design.

3.3 Repository

The repository for this project is hosted on GitHub and may be found here. The repository was built for the purpose of making deployment onto a Raspberry pi with docker easy with a few simple commands. This is made possible by docker-compose as shown in section 3.5. The command for starting up the containers is:

- docker-compose docker-compose.yml up

3.4 OctoPrint

Octoprint acts as a proxy between the website and the printer. During the end phase of website development, we shifted focus to implementing a channel for communication with a printer. To accomplish this we used Octoprint to handle the backend components of the application such as keeping track of the printing process. Octoprint is vital because it is capable of keeping track of the queue of projects running while also allowing administrators to collect and swap projects by using simple REST API calls. OctoPrint has its own in-depth documentation which may be found at OctoPrint Documentation.

3.5 Docker compose

The file docker-compose.yml is crucial for the easy and complete deployment of both the website and OctoPrint. Figure 3.5.1 below shows a

snippet of the file in which the USB devices of the pi are exposed to the container to be used by OctoPrint. This step is one of only 3 ways in which the containers have external access to pi's operation. The device "ttyAMA0" is the interface in which OctoPrint physically connects to the printer and exchanges data.

```
services:
    octoprint:
        image: octoprint/octoprint
        restart: unless-stopped
        devices:
            - /dev/ttyAMA0:/dev/ttyAMA0
            - /dev/ttyUSB0:/dev/ttyUSB69
        volumes:
            - octoprint:/octoprint
        networks:
            - secureprinting
```

Figure 3.5.1: Defining what hardware to expose to OctoPrint

```
environment:
    - OCTOPRINT_URL=http://octoprint:80
    - OCTOPRINT_APIKEY=${OCTOPRINT_APIKEY}
    - SECRET_KEY=${SECRET_KEY}
    - ALLOWED_HOSTS=${ALLOWED_HOSTS}
networks:
    - secureprinting
```

Figure 3.5.2: Important environment variables for the website to communicate effectively with OctoPrint

Figure 3.5.2 shows where in the docker-compose environment variables are located and imported into the project. OctoPrint's key and url allow for the website to communicate with OctoPrint's REST API and subsequently direct operations of the printer. The list of allowed hosts is beneficial as it restricts access to the users that are already allowed to access the machine. This variable import is perfect for development however upon deployment it is highly likely that this value should be changed to accept more inbound connections.