Group 8 Portfolio

Ibrahim Alsaleh, Nicholas Rust, Max Weimer

26 November 2019

Project Demo

View the User Documentation Here

We are unable to provide a full working website at this time. However, the above links to the documentation as the website is intended to be used. This version does not look great, but this is the version that has all of the features working currently. In addition, even if the look does change, it is unlikely that the website usage will change much, only the screenshots will have to be modified. The link under the “Summary” section is to the same documentation. We intend on having a demonstration for the professor after break to show progress made.

Objective

The DIFM website itself was originally created in order to assist agriculture researchers in having easier access to the data, in addition this website should ease the burden on farmers of uploading all of their files for the researchers to look at.

Our objective with this project was to give the DIFM development website a new look, and perhaps make some additional enhancements as we went (uploads, downloads, etc.). One of our additional goals, time allowing, is to migrate the website to django.

Summary

View the User Documentation Here

Note: The user documentation is missing a couple of screenshots, and the current screenshots may be outdated, but as of today, the 11th of November 2019, this is the version that works.

The website utilizes flask, the box developer API, postgresql, SQLite, HTML, and javascript. Ideally the website would work as a gathering point/interface for both farmers and researchers to upload/download files they have stored elsewhere, in private box accounts, locally, etc. This method will also allow for many layers of redundancy, if one of them fails, say the farmer's local pc, we still have access to the files on box and the database.

We attempted to use Materialize in order to give the website a fresh new look. However, we discovered that the official website was planning to be hosted on a Windows server running IIS. We found that flask does not play well with IIS and needs to be haphazardly stapled on in order to get it to work. In addition, materialize implements and overwrites a lot of javascript, so the javascript which was implemented in the website previously stopped working. Therefore, due to these two issues we can either rewrite the javascript in a number of files to work with materialize, or port the application over to django, which has much better documentation.

We utilize flask to hold all the other parts together and create forms with the wtf_flask library extension of wtform. With this we were able to use the box sdk in python to link to box and sqlalchemy to link with the database. One incredibly helpful resource we found was this Mega-Tutorial, it covers basically everything one would need to know to have a nicely working flask website.

The backend utilizes the box python sdk to connect with box. We set up OAuth2 with a secret key in order to connect to an internal storage account specifically for the application, This allows all users with an account to log in and be authorized to upload files to the box directory.

Member Page

It should be noted that we all spent a number of hours “practicing” the website setup steps and testing the existing setup documentation on two different linux operating systems and one Windows 10 system. Two of the three setups were successful and the setup documentation works well for Windows but can use some tweaks for Linux, in addition, database setup needs to be added to the documentation as it is not entirely straightforward. If we port to django the entirety of the documentation will have to be redone.

Ibrahim Alsaleh

Before I started working on the project, I had to figure out how to set up the DIFM website. After I downloaded the entire project from Bitbucket, I read the installation instructions file that was provided on the project folder and then I started to download the requirements for the website like (Anaconda, postgresql). I struggled with some steps because it’s written for the Linux OS and I am using Windows OS. So, it took me some time to get it working on my machine. Then I attempted to help Nicholas on css for styling the website and make it look better. We tried to style some of the website buttons, but that didn’t work with Flask because we noticed that all the css files should be saved in a different path in order for Flask to recognize it.

Nicholas Rust

I’ve been working on the project for a while, doing backend flask, javascript, html, and working with the box-api. For this part of the project I learned the structure of materialize after Max implemented the basics of it. Ibrahim and I then worked on getting the submit type buttons looking halfway decent using css. We struggled with the css because it goes in a different folder and needs specific path information to work with Materialize and Flask, so that took us some time. In addition, I began porting some of the website backend (specifically the databases) into django and working through a website setup tutorial. I read a decent amount of general django documentation and django documentation regarding postgresql databases. It seems the port may take a bit more time than we have, but at least we can get it started.

Max Weimer

The original plan was just to skin the existing backend and leverage mostly, if not only, css to make the necessary changes. I had a passing interest in Materialize, because it looks really nice and Google had been doing a push for their stack to look more like it. When I began working on the project I wasn’t very familiar with flask, but I had a lot of experience with Django from a previous class project, and quickly noticed the amount of similarity. After talking over the requirements a little bit it seemed like the jump to django wouldn’t be too bad, and would better suit the scope and end deployment scenario. Due to time constraints and other events I didn’t get to contribute as much to the django port as I would have liked within the time frame.

Conclusion

Having a bit more time to organize the group would have been Ideal, additionally communication within the group was lacking. If we were to restart the entire project, ideally we would have chosen a web framework that works well with IIS. We all learned about Materialize, Flask and a bit about django. Realistically, I would say that Flask did not work, and the unnecessary javascript implemented in the project was broken by Materialize. The javascript was unnecessary because forms can do the same thing it was implemented to do, we basically attempted to reinvent the wheel.

The project already having been started quite some time ago made it difficult for the new group members to join in and be effective right off the bat, it took a good chunk of the time for everyone to learn how the website was set up and figure out all the quirks. This lead to a bit of time where we were unable to really work on the project and instead were just testing things to get it to run and make sure everyone understood the layout.

References

-- A bibliography or listing of work used in the project or cited in the paper

- **Postgresql**: https://www.postgresql.org/
- **Flask** : http://flask.palletsprojects.com/en/1.1.x/
- **Flask Mega-Tutorial**: https://blog.miguelgrinberg.com/post/the-flask-mega-tutorial-part-i-hello-world
- **Box developer API**: https://developer.box.com/
- **SQLite**: https://sqlite.org/index.html
- **Materialize**: https://materializecss.com/
- **Django**: https://www.djangoproject.com/
- **Django Tutorial**: https://docs.djangoproject.com/en/2.2/intro/tutorial01/