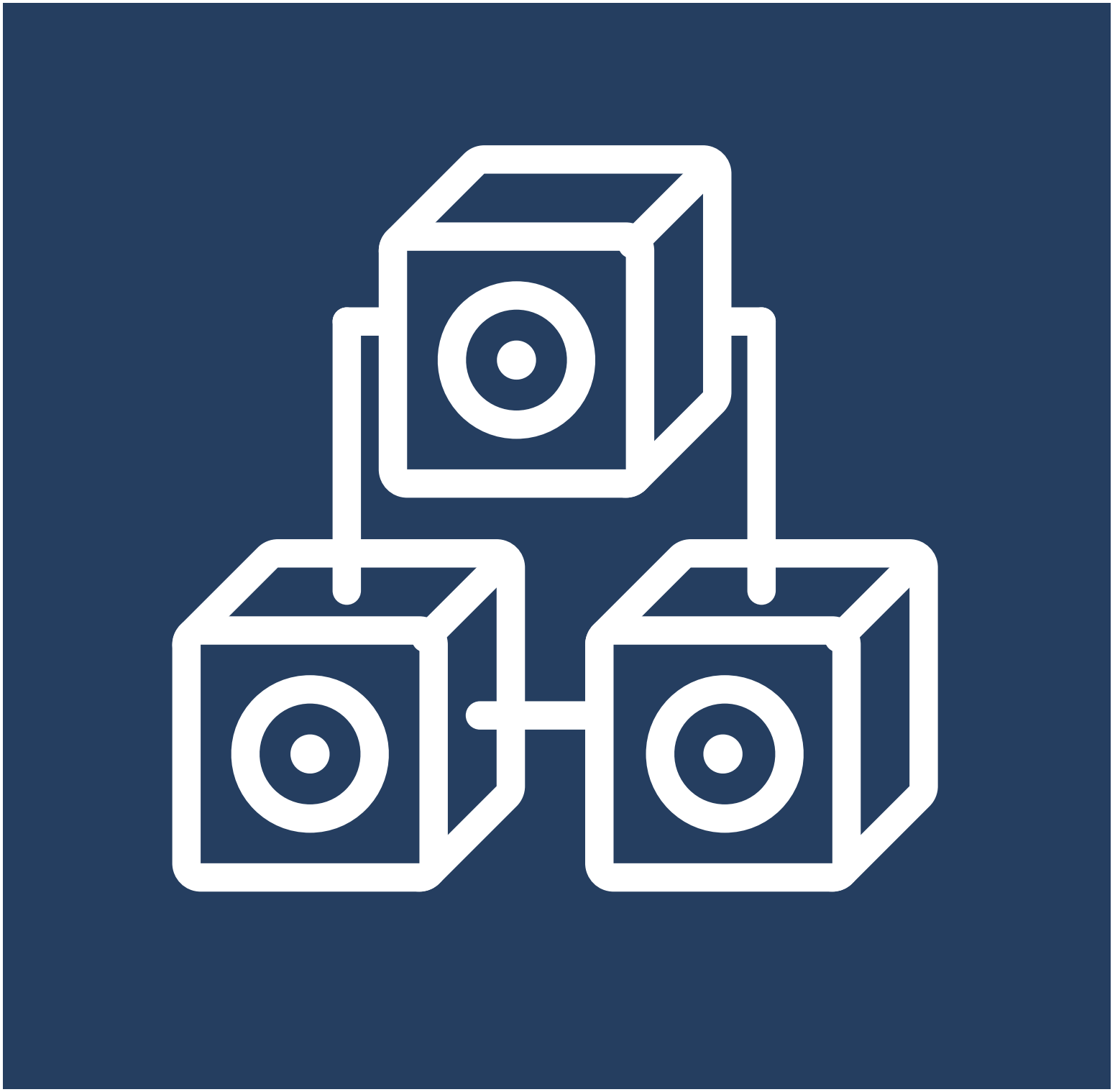
CONTENT SHARING VIA BLOCKCHAIN

MIAMI UNIVERSITY
CSE 274
HONORS COURSE EXTENSION
28 NOVEMBER 2020

BY KIM T. HA
HATK@MIAMIOH.EDU



INTRODUCTION

Blockchain

Blockchain is one of the most popular emerging technologies in modern time. The mechanics of blockchain is not that hard to understand either.

The idea behind it is, in simple words, to store records of data in blocks and link them together by giving each block an unique key, which is created using the data that it stores, and the reference key of the previous block, which acts similarly to a pointer and is the link in a chain. This makes the chain especially secured because when someone try to change the information of one block, the unique key of that block will be changed and, the block before it will no longer store the valid reference key. By the way information is stored in a blockchain, any changes to the chain will not go unnoticed.

One of the most popular applications of this technology is Bitcoin which now has millions of users. Other than cryptocurrencies, blockchain can also be applied in voting, logistics, or data storing and sharing.

This project and paper

This project will explore one of the uses of blockchain technology, which is content sharing. The aim is to control copyright and rightfully identify the creators of contents. What I have created here is definitely not an innovative platform that changes the world, but rather a foundational and simple web application that helps me learn about the technology, the framework and how to put together a web application.

The idea was given to me by my professor, Michael Stahr, who had a student previously worked on the same topic. The project is based and build on a tutorial by

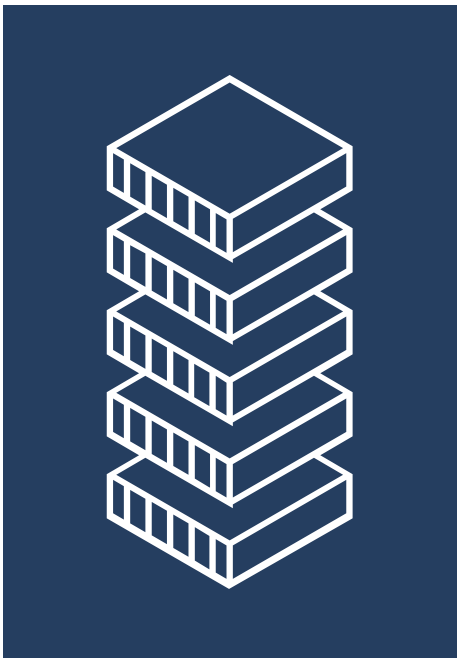
This paper will not be a comprehensive directions of how to build everything from scratch, but rather an outline of the project and my experience. For a complete tutorial, please refer to the links at the reference page.

GENERAL INFORMATION

Technology Stack

This project used Python as the back end language, Flask Micro-framework, and Scripting languages, which are HTML and CSS

- Python is used to write the structure of a block and a chain.
- Flask is the micro, which means it is easy to learn and use, web framework. It acts as the bridge between the backend and frontend parts of the project.
- HTML is responsible for the content of the page
- CSS: the representation of the look page

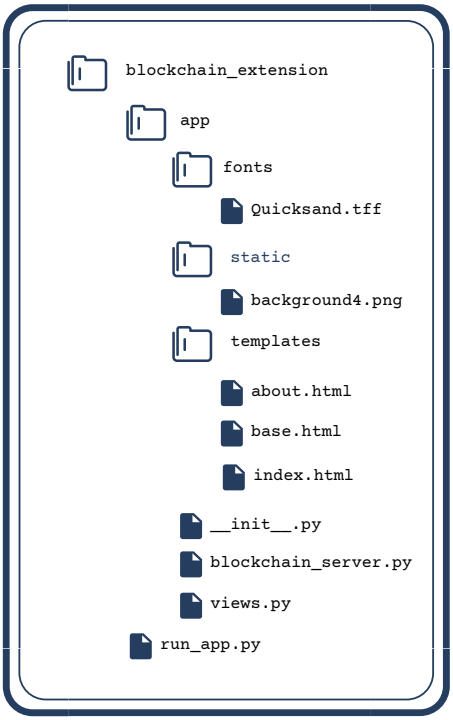


Directory Structure

The directory is separated into two parts:

- The app folder that has all the details about the implementation of the web application
- The run_app.py file, which will be the file that will be run but does not contain much information.

There are definitely other ways to organize the directory, and this is surely not the best way, for example, the fonts folder should be inside the static folder.



FILES

blockchain_sever.py

This file is the first to be created and also the foundational file of the entire project. In this file, we have three important sections:

- Classes, which are basically the templates for behaviors (methods) and characteristics (fields) of future resulting instances of those templates (objects)
- Functions, which are the methods that are not associated with any classes.
  - Some of the methods are associated with a link, while some are not.
  -
- Libraries: This is basically like a normal libraries that contain previous works of different people that we can take advantage of.

Class Block and Blockchain

A block will have different features, such as the index of the block, the content, or the time it is was created, etc., and can compute the unique key and proof of work number, which here is a number that indicates some extended computation has to be done to create the unique key of the block since we want the block to be as secured as possible. All of these features and behaviors are stored as fields and methods, respectively, in the class Block. Similarly, the Blockchain class also has all the desired characters and behaviors stored as fields and methods in the body of the class. The fields will start with "self." and the methods will start with "def".

Functions

Functions are like methods, which also start with "def", but not associated with any class - templates, and, therefore, are not behaviors. They are more like actions that we take to process certain information

Some of functions in the file have "@app.route('/link_sample', methods=['request_type'])" above (all the changeable parts are in green). This is to indicate that a certain function will be executed when the link provided is entered. The methods here are not the methods mentioned earlier but rather what the user is trying to do when entering the link (getting or sending - posting - information).

Libraries

Libraries here share a similar meaning with normal libraries, which both contain an enormous amount of works of different people. However, libraries in coding are digital and are reusable code that we can utilize. The reusable part is the same as referring to a book in a library to employ some of its information to solve a problem. In this file, there are different libraries are used, for example, "Flask" for using the Flask micro framework, "hashlib" to create the unique key for every block, or "time" to get the time. Libraries or files in libraries can either be "taken out" by using "import lib" or "from lib import file" (all changeable parts are in green and the line "from app import app" will be explained in __init__.py section).

views.py

This file only have functions which have been explained in **blockchain_server.py** section, and similar to the functions in blockchain_server.py, some functions here are also associated with links. The difference here is that the functions in views.py focus more on getting, sending the information to the appropriate functions in blockchain_server.py via links to process the information rather than doing the work themselves, and showing the right HTML files. The file also imports some libraries to help with the heavy work.

HTML files and templates folder

These are files that decide how the web application will look like. These files are written in HTML for the content, CSS for the representation, and Jinja to render user inputs and create reusable HTML files. (for more information, please refer to the Flask documentation). Some of the functions in views.py will say "render_template(...)", which automatically sends the indicated HTML file to the browser with all information. All HTML files will be automatically searched in the templates folder so there is no need to specify the entire path to the file.

__init__.py

This is the file that makes the directory into a package, another name for a library. This will make the importing process easier by not requiring specifying the path to

the file. All the objects (refer to the blockchain_server.py section) created can be used in all other files in the same directory. That is why in both files blockchain_server.py and views.py have the line "from app import app". The first app is the name of the library, which is made possible by the __init__.py file. The second app is the object created in the __init__.py file (This is not the best way in terms of naming, and can be improved by using different names for the library and the object).

run_app.py

This project can be run at the terminal by setting up the environment for the app and using the appropriate command lines while providing a port number. However, there is an easier way is to do everything mentioned above in a file, so the only thing that has to be done is to use the command line "python run_app.py" in the terminal. The file has the same line "from app import app" as explained above, and the "run" method (methods are explained in the **blockchain_server.py** section). The port number is also provided in the method. This is basically a way to wrap everything up in a file.

HOW TO RUN

- Right-click on the folder, go to "Properties" and copy the line after "Location: " and add the folder name to the end. It will look something like this:

```
C:\Users\user_name\Downloads\blockchain_extension
```

- Find and open Command Prompt on your computer and enter the following line cd + space + the line you found from the previous step:

```
cd C:\Users\user_name\Downloads\blockchain_extension
```

- Run the run_app.py file by using the following command line:

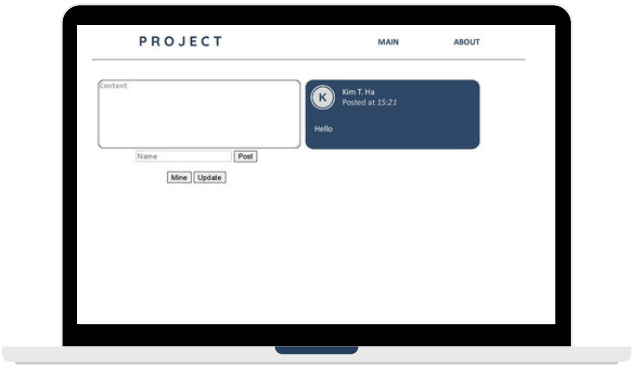
```
python run_app.py
```

- Wait for a minute and you will see a line that has an URL link (it;s likely that it will be the last line. Copy that link to your web browser to the see and use the app or you can go to http://.

```
Running on http://127.0.0.1:5000/ (Press CRTL+C to quit)
```

- After entering the link in a web browser, the page will appear (1). Enter what you want to post and your name. After that click "Post". Click "Mine" and you will be redirected to a page saying that the block is mined. To make your post appear on the right, go back to the original page and click "Update" (2).

1



2

REFERENCES

Kansal, S. (2020, January 29). Develop a blockchain application from scratch in Python. Retrieved November 28, 2020, from https://developer.ibm.com/technologies/blockchain/tutorials/develop-a-blockchain-application-from-scratch-in-python/

Templates. (n.d.). Retrieved November 28, 2020, from https://flask.palletsprojects.com/en/1.1.x/tutorial/templates/