Stamp the Web

Developer’s Documentation

Table of Contents

1. Introduction

This document describes the ‘Stamp the Web’ system in detail. Besides discussing functionalities, it highlights the technologies, modules and APIs used in the development of the system.

1.1 Purpose

This document is designed to help understand the technical aspects of the system. With the help of this document, any software developer can enhance the functionality of this system, or make improvements in the future, if required. It provides sufficient information for a programmer to produce the software, and for a maintainer, who may not be the developer himself, to make subsequent changes.

1.2 Scope

‘Stamp the Web’ allows its users to partake in securing digital heritage online. It is a trusted timestamping service for web-based content that can be used free of charge. The service enables users to automatically create trusted timestamps to preserve the existence of online content at a certain point in time, such as newspapers articles, Wikipedia pages, blog posts, etc. This enables users to prove that certain information online existed in a particular state at the time it was 'trusted timestamped' using ‘Stamp The Web’. It can be accessed online through https://www.stamptheweb.org. In this documentation ‘this’ or ‘the’ ‘system’ are both abbreviations for ‘Stamp the Web’.

1.2.1 Developing a tool for archiving and comparing news articles on the Web.

For scientists and information researchers we require a tool for efficient access to the web archived content[1]. Ease of use, extensibility and reusability is also important for a web archive system. The system should be fast and easy to use at the same time in order to encourage usability.

‘Stamp the Web’ stores the hashes, or for all intents and purposes the timestamps, of the online content independently. These hashes are sent to Bitcoin’s blockchain with the help of the ‘Originstamp’ API [2] for later verification from Bitcoin’s decentralized network. Thus, the user has the opportunity to verify timestamps of any online content from ‘OriginStamp’ as well as from the

[1] "Proceedings of the 16th ACM/IEEE-CS on Joint Conference on Digital ..." 2016. 23 Jul. 2016 <http://dl.acm.org/citation.cfm?id=2910896&picked=prox>
[2] "OriginStamp: Trusted Timestamping with Bitcoin." 2016. 23 Jul. 2016 <https://www.originstamp.org/>

Bitcoin network. ‘Stamp the Web’ also redirects to ‘OriginStamp’ for verification. Every timestamped content is associated with a hash value, which represents the online information that is supposed to be timestamped. This hash value is required in order to verify the timestamps from ‘OriginStamp. Some of the functionalities of this system are as follows:

- Stamp a web article, news article or blog post etc.
- Compare two saved (timestamped) web pages with each other or with the current online version.
- Search for other saved articles by text or by categories.
- Compare a timestamped article with the same article from any other part of the world e.g. Russia, UK, USA etc.
- Compare a timestamped article from a certain date with its current online version.
- Compare a timestamped article from a certain date with its timestamped version from another date.
- Check if an online article is blocked in any other part of the world.
- Check where in the world an article is blocked through a map.
- Review statistics of web pages most visited in the system with respect to country of origin.
- Timestamp an article according to a user-defined schedule. If any change is detected in that article, the user is notified by email (if email is provided), or information is automatically stored in the system.
- Timestamp an article according to a user-defined schedule and compare it at regular intervals with the same page from other countries (selected countries). The user is notified by email if changes are found.
- Account actions: Manage your account, passwords, email, profile, etc

1.2.2 Showing similar records together

Corpora building is a fundamental task for a Web Archive system [3]. It means showing similar posts, results together in a meaningful way. We show similar results from posts together, when a keyword is searched. Same categories are kept together in ‘browse other submissions’. We make categories by using domain names. In order to efficiently use information saved, we are using a temporal view whenever information is searched for or browsed through. We provide easy access of archived data to political research scientists by dividing content into categories and showing timelines with different views.

‘Stamp the Web’ serves many purposes, it can for example serve as a web archive, online heritage protection system or as a web articles aggregator. There are many existing

[3] Holzmann, Helge, Vinay Goel, and Avishek Anand. "ArchiveSpark: Efficient Web Archive Access, Extraction and Derivation." *Proceedings of the 16th ACM/IEEE-CS on Joint Conference on Digital Libraries* 19 Jun. 2016: 83-92.

applications for similar purposes (i.e. web heritage protection) such as archive.org [4], Google news archive [5] CNN news archive [6] BBC news archive [7] etc. While these sources can somehow protect information heritage on the web, they are not secure, because information could be hacked, manipulated by administrators, or there is a possibility of server failure. This system protects online web heritage, as well as offers the tamperproof timestamping of online content, in particular, online news coverage or other information of the user's interest. In the Internet Archive one can archive any web content for the future to see, but it does not add information to the Bitcoin network to enable the verification of the content. In contrast with 'Stamp the Web', our aim is to provide a secure web archive with the possibility to verify the information. Thus 'Stamp the Web' presents a system with a strategical advantage and better security mechanisms in terms of verification and transparency.

1.3 Definitions, Acronyms and Abbreviations

WSGI: The **Web Server Gateway Interface** (**WSGI**) is a specification for a simple and a universal interface between web servers and web applications or frameworks for Python. [8]

Satoshi: A Satoshi is the smallest fraction of a Bitcoin that can currently be sent: 0.00000001 BTC, that is, a hundredth of a millionth BTC.[9]

Flask Blueprint: A blueprint defines a collection of views, templates, static files and other elements that can be applied to an application.[10]

Jinja2: A templating language for HTML documents, to enable inheritance in HTML documents and thus increases the speed of development and reduces redundancy in HTML.

[4] "Internet Archive: Digital Library of Free Books, Movies, Music ..." 2006. 23 Jul. 2016 <https://www.archive.org/>

[5] "Browse all newspapers - Google News." 2011. 23 Jul. 2016 <https://news.google.com/newspapers>

[6] "CNN Student News - Archive - CNN.com." 2015. 23 Jul. 2016 <http://www.cnn.com/specials/student-news-transcripts>

[7] "BBC Archive." 2007. 23 Jul. 2016 <http://www.bbc.co.uk/archive/>

[8] "Web Server Gateway Interface - Wikipedia, the free encyclopedia." 2011. 23 Jul. 2016 <https://en.wikipedia.org/wiki/Web_Server_Gateway_Interface>

[9] "terminology - What is a 'Satoshi'? - Bitcoin Stack Exchange." 2011. 23 Jul. 2016 <http://bitcoin.stackexchange.com/questions/114/what-is-a-satoshi>

[10] "Blueprints — Explore Flask 1.0 documentation." 2016. 23 Jul. 2016 <http://exploreflask.com/en/latest/blueprints.html>

2 System Overview

The system has been designed using Flask. Flask is a new python web framework in comparison to other python web frameworks like Django, Pyramid etc. As Flask is a micro web application framework and does not require large frameworks in comparison to Django or Pyramid, it has the tremendous ability to help develop applications faster. Even so, Flask applications are extensible[11]. Flask uses Jinja2, which is a templating language for python. Thus working on the front end of 'Stamp the Web' requires basic understanding of Jinja2. Flask is based on 'Werkzeug' which is a Web Server Gateway Interface(WSGI) library for python.
Due to the above-mentioned advantages, Flask is found to be suitable for creating this application. It has different packages that support sophisticated applications development. Besides using Flask, the application is tested in python 3.4.2 and upwards as well as for python version 3.5.2. On the deployment server on Amazon EC2 we are using python 3.4.3. Python 3 was the logical choice as python 2.7 support is only available for 4 more years, since 'end of life' for python 2 is in 2020. Hence using python 3 is a good option for the future.
Python has tremendous support for remote operations, such as accessing web pages from different remote locations, for example through proxies, and therefore it also was a good option for 'Stamp the Web'. For instance, the proxy features of python help us find out if a web page is blocked in another location in the world and they help us find out locations of web articles visited. Only free python libraries have been used in the entire project. For better understanding of the python packages used, all the different packages used for specific operations are discussed later in chapter 5.

2.1 System Characteristics

'Stamp the Web' uses secure HTTP i.e HTTPS to be accessed through the web. It is attempted to use global variables throughout the system for different configurations. If these variables are stored into the operating system, the configuration of the system becomes easy. Thus it is easy to append new features into the system. It just requires changing global variables or adding new global variables. For example switching to another database requires a change in the following environment variable:

```
DEV_DATABASE_URL = <url for the DB>
```

The system first reads a variable from the environment variables of the operating system if available, otherwise it takes default values from the code. For good practise, environment variables should be configured. The following commands add global variables to the heroku web server:

[11] "Foreword — Flask Documentation (0.11)." 2014. 23 Jul. 2016 <http://flask.pocoo.org/docs/latest/foreword/>

```
$ heroku config:set MAIL_USERNAME=<your-gmail-username>
$ heroku config:set MAIL_PASSWORD=<your-gmail-password>
```

In the server setup the command is the following:

```
$ MAIL_USERNAME=<your-gmail-username>
$ MAIL_PASSWORD=<your-gmail-password>
```

Or the variables can be set in ~/.bashrc directly.
The provided email account through these variables would be used to send emails from the system, for example new registration, account confirmation etc. Furthermore, different types of configurations are available depending upon which environment the system has been started with, i.e. Development, Testing or Production.

2.1.1 Login and Security

The Flask-Login module has been used to make login sessions secure. Using 'strong' protection, the application automatically logs users out, if the IP-address of the user has changed. Furthermore, a token based authentication scheme has been used with a very short expiration time(i.e 3600 seconds) for each token. The client must send the authentication token with each request to the server in order to authenticate himself. As soon as a token expires, the client must re-authenticate to the system to get a new token. Flask's module 'itsdangerous' has been used to generate and validate the tokens. A user must provide an email in order to register. The user receives an email in order to activate the account and in order to start using the system. A strong password containing at least 6 characters is required for creating an account.

2.2 System Architecture

The system does not use any specific architecture, however in order to efficiently manage application code, different blueprints have been used. The system's front end files have been separated from script and backend files. We are using Flask and it is a micro web framework. However different modules have been separated in order to support organization of the code. Flask also supports blueprints and so two blueprints have been used. Flask blueprints are ideal for larger applications, in order to factor applications into set of blueprints. [12] Thus figure 1 depicts the component view of the system based on the two blueprints (main_blueprint, auth_blue_print). If a request arrives, related to authentication such as login, registration, authentication, it is handled using the 'auth_blueprint'. Other requests are handled by the 'main_blueprint'.

[12] "Modular Applications with Blueprints — Flask Documentation (0.11)." 2014. 23 Jul. 2016 <http://flask.pocoo.org/docs/latest/blueprints/>

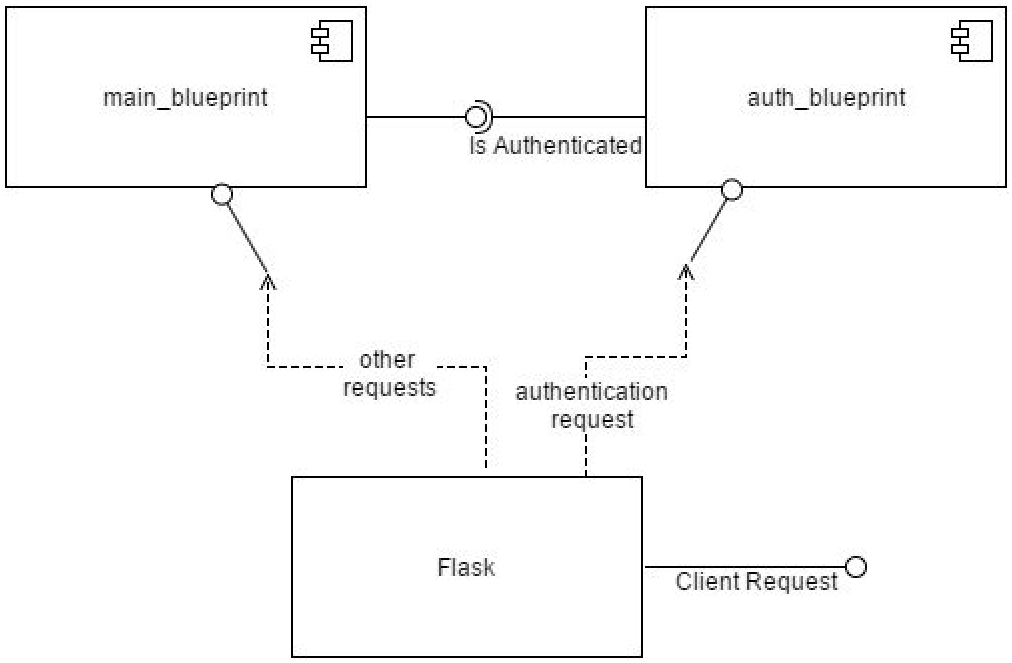


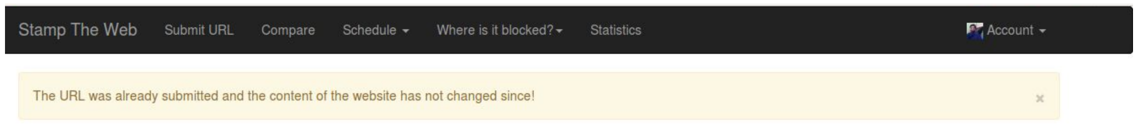
Figure 1: Component diagram ‘Stamp the Web’

Both these blueprints contain different set of files, folders and other blueprint specific documents.

2.3 Infrastructure Services

Utmost care has been taken during code-writing so that whenever an input or output operation is performed, the application should be able to handle an exception that occurs during the process. The errors messages and information messages are reported in a sophisticated way using ‘Flash’ error messages by Flask. Thus if an unwanted error occurs the user should be able to report, or at least user may know what is going on. A sample ‘Flash message’ is shown in figure 2. One example code for showing Flash message to the user as well as handling exceptions is given below:

```
try:
   pdfkit.from_url(url, path)
except OSError as e:
  if not app.config["production"]:
       flash(u'Could not create PDF from ' + url, 'error')
```



The URL was already submitted and the content of the website has not changed since!

Figure 2: Flash message in ‘Stamp the Web’

Similarly in order to support developers and testers, background logs and information are collected. If any error or any other activity is done it is logged into the ‘webStamps.log’ file, which is present at the root directory of the source code. Not only error messages, but also information messages are recorded throughout the system. An example from ‘webStamps.log’ file is shown below, where it indicates that a new PDF file with the given name has been created into the system. Later, a new Post has been added into the system with the date and time at which this was done.

```
‘webStamps.log’
But local PDF exists at: app/pdf/QmZsoGMPq7Jr1gVtJzojqaR6gkKmRMSmTKQysF1epiQYDJ.pdf
2016-07-27 17:25:12 New Post added
```

It is a good practise to report errors and other information happening in the system into logs. The sample code for logging errors and information into the logs is as follows:

```
current_app.logger.info(datetime.now().strftime("%Y-%m-%d %H:%M:%S") + " New Regular task added")
current_app.logger.error(datetime.now().strftime("%Y-%m-%d %H:%M:%S") + " An error occur")
```

The object ‘current_app’ could be used to get the flask ‘app’ and app has already started a ‘logging handler’ into the ‘init.py’ file, at the time of application creation. Thus we can simply use ‘logger’ anywhere in the code.

3. System Context

‘Stamp the Web’ uses ‘OriginStamp’ API [13] to generate timestamps from the hashes that correspond to online content. We consider only text from the provided web articles and exclude pictures, ads, links, etc. ‘OriginStamp’ which uses aggregated hashes once a day for Bitcoin’s transactions and only generates minimal Bitcoin transaction fees, i.e 1 Satoshi. In return ‘OriginStamp’ API provides us the timestamp against the provided hash. Although the hash is still not sent to Bitcoin blockchain. Hence at this point the hash will not be verifiable in the Bitcoin network yet. It requires one day for ‘OriginStamp’ to collect all the hashes and send them to the Bitcoin block chain.

Bitcoin is a widely used crypto-currency and it is found to be secure so far. It was introduced in 2008 [14] and it has the highest participating nodes on the internet. The larger number of nodes is considered to be a security feature as discussed by the creator of this system. It is impossible to generate a Bitcoin attack unless the attacking nodes exceed 50%. Furthermore Bitcoin also uses a decentralized blockchain to store transaction information.

This transaction information is tamper proof due to the use of block chains from the Bitcoin network. Furthermore, tampering would require a change in the whole transaction tree (based on the idea proposed by Merkle in 1988 [15]) as the most recent transaction contains the hash of all previous transactions. In order to make this service tamper proof, Bitcoin block chain has a sophisticated method of storing this information as a transaction by forming a hierarchical architecture by using merkle tree and nonce in such as way that changing any information would be quite impossible. Also, Bitcoin stores information of all the parent transactions providing it with another layer of security and tampering with any information needs a change in all upper levels of the transaction tree. To find more about ‘OriginStamp’ and Bitcoin block chain please see review. [16]

‘Stamp the Web’ simply sends hashes of the requested web pages to ‘OriginStamp’ using python requests. In reply ‘OriginStamp’ sends the timestamp and related information using JSON format. The sample code for generating timestamps is shown below:

```
headers = {'Content-Type': 'application/json', 'Authorization': 'Token token="<Token
To Be Generated from API"'}
data = {'hash': hash_variable, 'title': title_of_web_page}
requests.post(api_url, json=data, headers=headers)
```

[13] "Developer resources - OriginStamp." 2016. 6 Aug. 2016 <https://www.originstamp.org/developer>
[14] Nakamoto, S. "Bitcoin: A Peer-to-Peer Electronic Cash System." 2014. <https://bitcoin.org/bitcoin.pdf>
[15] Merkle, Ralph C. "Fast software encryption functions." *Conference on the Theory and Application of Cryptography* 11 Aug. 1990: 477-501.
[16] Gipp, Bela, Norman Meuschke, and André Gernandt. "Decentralized Trusted Timestamping using the Crypto Currency Bitcoin." *arXiv preprint arXiv:1502.04015* (2015).

In response of the above code a JSON value is returned and from that 'Date' field contains the timestamp information of the provided hash.
Figure 3 shows the deployment view of 'Stamp the Web' and describes how it is operating with other systems.

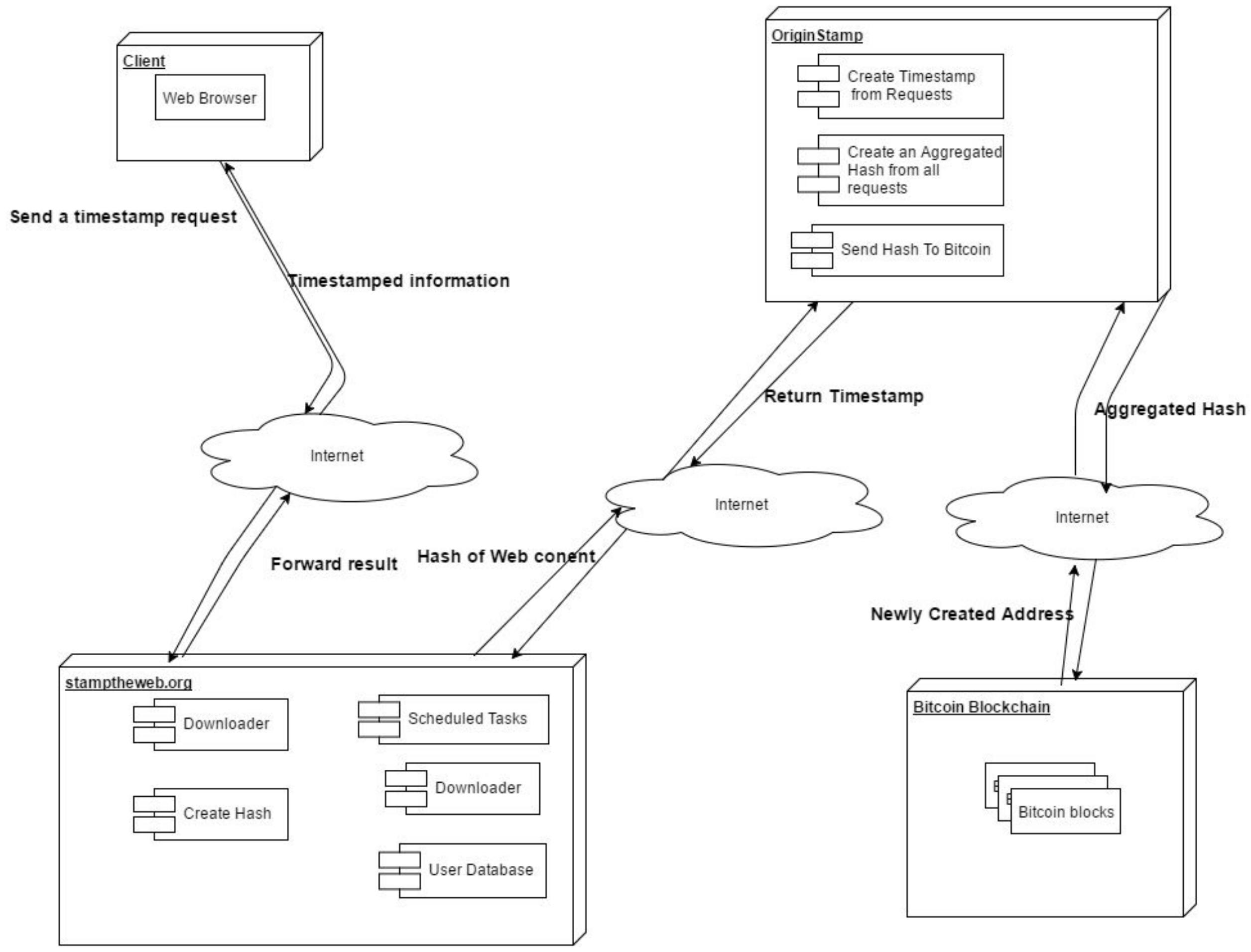


Figure 3: Deployment view of 'Stamp the Web'

3.1.1 Technologies Used

A range of technologies have been used in the development of front end of 'Stamp the Web'. It is attempted to use most of the technologies online by providing the URL into the source. Some of the used technologies are as follows:

Jquery
Javascript
Bootstrap
JSON

D3
Ajax

There are different files have also been used on the server locally, for example javascript and jquery files. There are some D3 supported maps in the system, which require SVG images into the HTML files. The coordinates of these SVG images have been read using JSON files. These files also have been changed according to the user instructions into the python code at the server side. Some new JSON files have been created, depending on the specific requirements and instructions given by the user. Tab separated and comma separated files have been used for reading some data like proxies of different countries in order to open a URL from different locations.

Besides using general purpose technologies, some code for special purposes have been used, for example bootstrap calendar [17]. This calendar is used to show the calendar view of different timestamped pages, either by the result of a search or through selected domains from the system. There is also an active community for this calendar, any help or bug fixing could be found from github [18]. Moreover ‘simple map’ in D3 [19] modules have been used for creating maps for statistics, locations of proxies and showing block map of a provided URL.

[17] "Twitter Bootstrap jQuery Calendar component." 2013. 7 Aug. 2016 <http://bootstrap-calendar.azurewebsites.net/>
[18] "Issues · Serhioromano/bootstrap-calendar · GitHub." 2013. 7 Aug. 2016 <https://github.com/Serhioromano/bootstrap-calendar/issues>
[19] "Simple Map D3 - MinnPost." 2013. 7 Aug. 2016 <http://code.minnpost.com/simple-map-d3/>

4. System Design

If we divide both blueprints as discussed above in the system architecture into two different modules then some of the used sub modules and their dependencies are explained in the module view of the system. Since the system is under development, this diagram may change with time. As the rule of a module view diagram, a dotted line represents the dependencies on other sub-modules, as shown in figure 4. In some cases it would not be easy to find code for some modules. However this division divides the sub-tasks into a diagram. To help understand different set of functionalities into the system, each individual module is explained as follows:

Login: Contains all the activities related to logging in the user.
Confirmation: Confirmation activities such as confirmation token, and confirming after the email verified.
Registration: Registration activities such as sending email for completing registration.
Management Functions: Account management functions such as change in email, change in password etc.

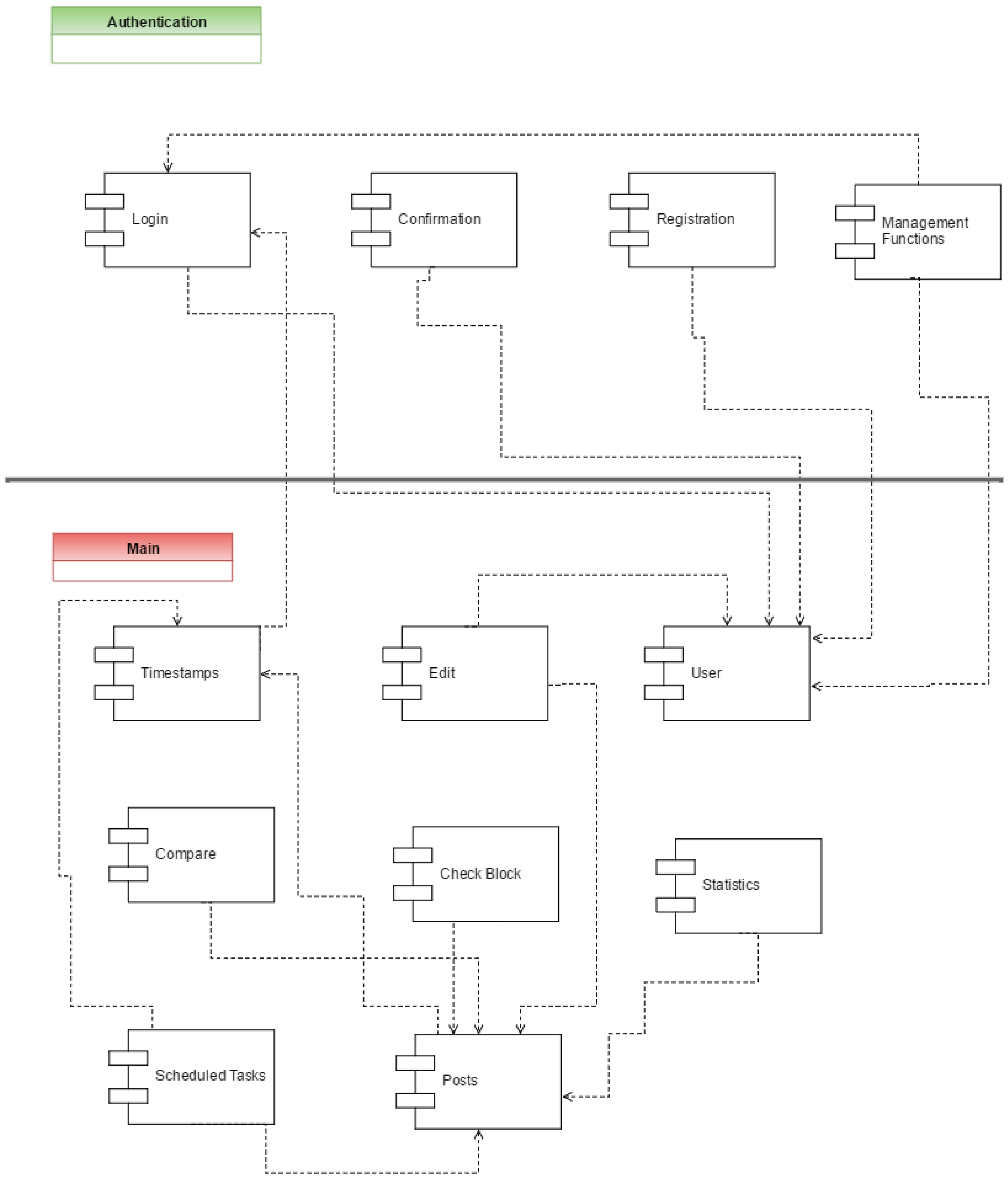


Figure 4: Modular View ‘Stamp the Web’

Timestamps: Creating timestamps from the text in the URL with the help of ‘OriginStamp’. It also contains creating pdf and screenshots.

Edit: Edit operations for the articles and user.
Users: All the user related activities.
Compare: compare operations for different posts (with different countries or with each other).
Check Block: To check if post is block and all other block operations for selected articles.
Statistics: Activities related to create overall statistics visualization.
Scheduled Tasks: Activities which needs to be performed with a regular interval of time, i.e after every 86400 seconds.
Posts: Activities related to posts.

4.1 Database Design

In order to efficiently store data in a database, the database is divided into many tables and tables have been normalized to help prevent data redundancies. Database design of 'Stamp the Web' is shown in figure 5. Each table is illustrated in detail as follows:

location: This table stores the locations of web urls. First we get the public IPs of the URLs visited in the system, and then from public IP we obtain locations (by using free geoip[20]) in order to visualize the overall statistics. Since it takes some time to take location from an ip, hence if the location is already present in the location table the query is not sent to the free geoip API but rather obtained locally in order to save time.

posts: The table 'posts' contains all the information of a timestamped page. If a web page is requested to be timestamp again, the system checks the hash field in this table. If the hash is already present in the table, means the URL is provided again and there is no change in the content of the provided URL. If hash is not present it means a new URL is provided or there are some changes in the content of the provided URL. System does not create multiple entries of an article, as it creates redundancies in the database.

regular: regular table corresponds to the scheduled tasks. It is in 'one to many' relationship with the posts, table. Since one posts can have multiple scheduled tasks, may be by different users. Again at the time of a 'regular' timestamp creation if a hash is present in the system just another entry is created in the regular table. A record in the 'posts' table is not created.

block: This table contains records of articles which are blocked in the selected countries. It is with 'one-to-many' relationship with posts. One post can be blocked in many countries.

[20] "Freegeoip.net." 2013. 26 Jul. 2016 <https://freegeoip.net/>

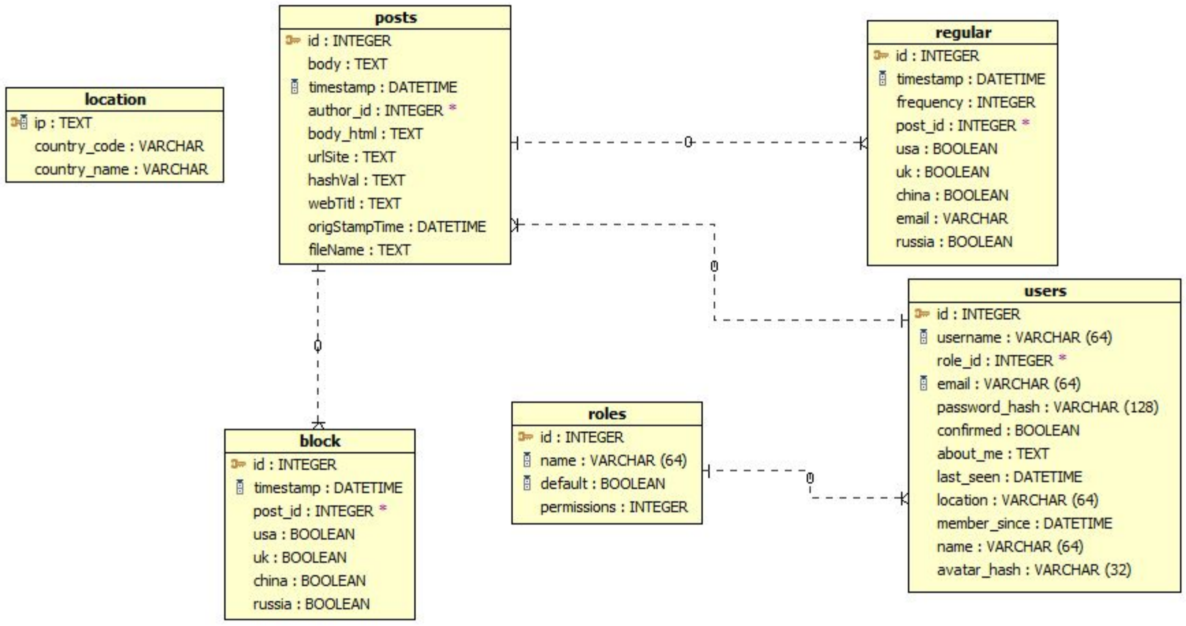


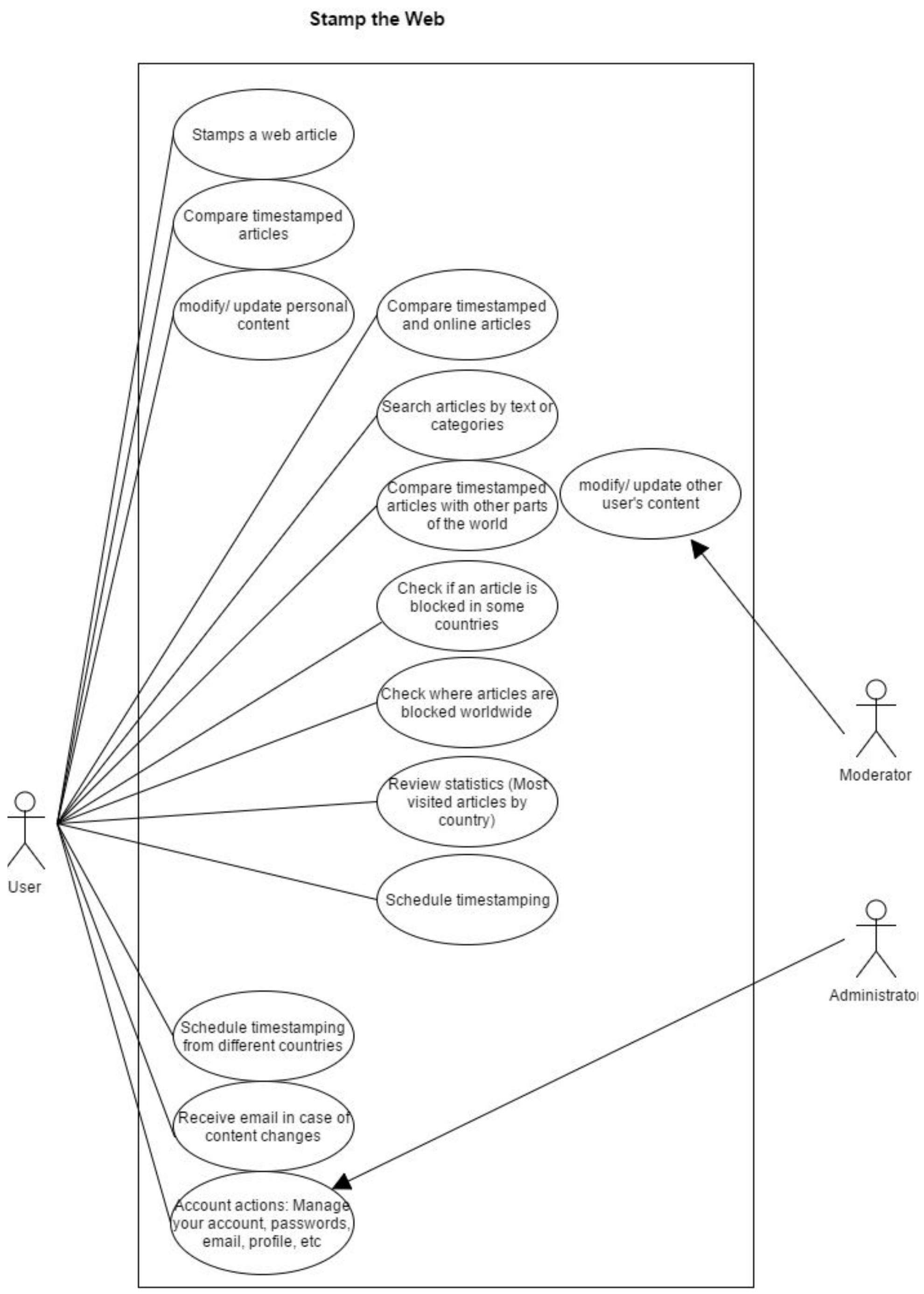
Figure 5: Entity relationship diagram ‘Stamp the Web’

roles: roles table contains information regarding roles for a user. There are three roles added in the system Administrator, Moderator and User. As soon as a user creates an account a ‘User’ is automatically assigned to the new user. The other two types of account are for maintenance purposes. However the roles can be extended. The roles table is also one to many relationship with users table, since one user can have many roles. The roles are described in details in figure 6 which explains the use case diagram.

users: the ‘users’ table contains information of all the users in the system. It is with one to many relationship with posts. Since one user can have multiple posts. As discussed earlier a post is a timestamped web article in the system.

4.2 Use Case Design

Every user registered in the system is a simple user. The other roles are just for configuration and administration. As shown in figure 6 use case diagram. A user which is registered can do everything which is available in the system. However a user can only update her/his personal information. In order to change information of other users (such as profile, account) an administrator user is required. Similarly a ‘Moderator’ can also do what a user can do. Moderator can also modify the content of other users such as posts. Moreover in future if other features are introduced in the system, such as comments, then a moderator may also be able to change, update or delete it or content of other users. A ‘moderator’ cannot update account information of a user only an administrator can do.


Stamp the Web
Stamps a web article
Compare timestamped articles
modify/ update personal content
Compare timestamped and online articles
Search articles by text or categories
Compare timestamped articles with other parts of the world
modify/ update other user's content
Check if an article is blocked in some countries
Check where articles are blocked worldwide
Review statistics (Most visited articles by country)
Schedule timestamping
Schedule timestamping from different countries
Receive email in case of content changes
Account actions: Manage your account, passwords, email, profile, etc
User
Moderator
Administrator
Figure 6: Use case diagram

4.2.1 User Roles

The user roles are stored in the 'roles' table. The 'roles' table contains permission column which defines each individual role of a user. This column contains a number which would be read as binary string of 8 digits. Each role is set according to this permission number. For example an administrator has all 8 bits on, since it has access to everything in the system. Similarly a moderator has the eighth binary bit on that indicates access to that content made by other users. Finally a normal user which is registered through registration in the result of 'sign up' it has the fourth binary bit on. The fourth bit indicates write access in the system. This is shown as follows:

Administrator:	11111111
Moderator:	00001000
User:	00000100

These roles are stored in the database as their decimal equivalent in 'Stamp the Web' as follows:

Administrator:	255
Moderator :	15
User:	7

For each user it is required to assign the 'role_id' the primary key of 'roles' table. In order to assign that user that corresponding role. The 'user' role is by default assigned to the user which has 'role_id' as 3. User roles have been already defined in the code, in 'Roles' class present in 'models.py' file. In order to transfer user roles from code into the database for efficient usage. It is required to run the following command from command prompt.

```
(venv) $ python manage.py shell
>>> Role.insert_roles()
```

In order to view all the roles present.

```
>>> Role.query.all()
[<Role u'Administrator'>, <Role u'User'>, <Role u'Moderator'>]
```

This way we can also define more roles by adding them into 'insert_roles()' method. This idea of setting up roles has been taken from the flask web development book [21]

4.3 Documentation Standards

It is attempted to arrange files and code in a reasonable manner. There are vast range of files have been used, the most prominent are python files '.py' and html files '.html'. Fo;es have been arranged according to the module they belong to. If a backend scripting file is a part of 'main_blueprint' it would be present in that folder. Moreover html files are present in 'templates'

[21] Grinberg, Miguel. *Flask Web Development: Developing Web Applications with Python*. " O'Reilly Media, Inc." 2014.

folder. There are further sub-sections created inside ‘templates’ folder. In order to analytically divide the files in appropriate folders. For example templates for sending emails to a user in case of change in content are in ‘templates/mail’ folder. Moreover configuration and management scripts have been placed in the ‘root’ of the application. The code is self explanatory and names of the files indicate the functionality of the files. Moreover comments have been used with the classes and methods in order to help other developers understand the code.

4.4 Naming conventions

Python naming convention have been used i.e PEP8[22]. It is encouraged to use variables methods and class names according to python naming conventions for future or as follows:

```
a_new_variable = 0
a_new_method():
class NewClass():
  def __init__(self):
```

4.5 Programming Standards

PEP8 codes style has been used while writing the code of python for server side scripting. The variables, methods and classes names follow pep8 naming conventions. Auto formating has also been done using ‘Pycharm’ as well as PEP8 has been used in ‘Pycharm’ to automatically indicate if some code is not according to the coding standards of python.

4.6 Software Development Tools

It is encouraged to use ‘Pycharm’ or any other python integrated development environment (IDE). In order to better manage code and different set of files included in the project. The backend code is easily debugged using an IDE, however some code is also present in the form of Javascript and Jquery which requires a modern web-browser, like Mozilla Firefox or Google Chrome to debug. Moreover templating language Jinja2 also contains some scripting, however it is understandable code and does not require any debugging. The application was initially developed using normal text editors, later ‘Pycharm Community Edition’ is used, which really help manage the code, and makes development fast.

-

[22] "PEP 8 -- Style Guide for Python Code | Python.org." 2014. 30 Jul. 2016 <https://www.python.org/dev/peps/pep-0008/>

5. Component Description

Flask framework makes the development of the system very easy, but it is still required to have good understanding of the different set of used components. Some of the used modules are listed below:

5.1 Flask-SQLAlchemy

SQLAlchemy is the Python SQL toolkit and Object Relational Mapper [23]. It helps application developers writing sophisticated code without writing SQL statements. The greater advantage of using it is not to worry about the backend databases. As soon as configuration parameters have been changed the SQLAlchemy starts using the databases whose configurations have been provided. Moreover it is just required to change the model of the database from the code. SQLAlchemy will itself manage the model of database and communicate with database with the new model. The programmer does not require to change the database design from the database itself. Similarly in our case it is required to change the 'model' file present at the app folder of the application. For example if it is required to change the 'users' table's model in the database. The following user class requires change in the code. Everything else would be managed by SQLAlchemy.

```
class User(UserMixin, db.Model):
   __tablename__ = 'users'
   id = db.Column(db.Integer, primary_key=True)
   email = db.Column(db.String(64), unique=True, index=True)
   username = db.Column(db.String(64), unique=True, index=True)
   role_id = db.Column(db.Integer, db.ForeignKey('roles.id'))
   password_hash = db.Column(db.String(128))
   confirmed = db.Column(db.Boolean, default=False)
   name = db.Column(db.String(64))
   location = db.Column(db.String(64))
   about_me = db.Column(db.Text())
   member_since = db.Column(db.DateTime(), default=datetime.utcnow)
   last_seen = db.Column(db.DateTime(), default=datetime.utcnow)
   avatar_hash = db.Column(db.String(32))
   posts = db.relationship('Post', backref='author', lazy='dynamic')
   very = db.relationship('Verify', backref='author', lazy='dynamic')
```

[23] "SQLAlchemy - The Database Toolkit for Python." 2005. 31 Jul. 2016 <http://www.sqlalchemy.org/>

Integrating SQL scripts into the code is very easy. It is required to use the built in SQLAlchemy methods in order to perform SQL operations into the code. There is no need to write raw SQL statements. Given are few examples which may be later used:

Adding a new record in a model class.

```
new_record    =    Model_Class(value_1=<value_1>,     value_2=<value_2>
…………,value_n=<value_n>, foreign_key=<foreign_key_object>)
db.session.add( new_record )
db.session.commit()
```

Querying from the database

```
result = Model_Class.query.get_or_404(search_value)
```

Performing SQL operations (AND)

```
result=
Model_Class.query.filter(and_(Model_Class.column_1.like(value_1),
Model_Class.column_2.like(value_2))).first()
```

5.2 Flask-Migrate

Flask migrate help change the database structure without affecting the existing data present in the database. After the change in the database model, it is required to let Flask-Migrate upgrade the database and effect the changes in the ‘mode’ into the database. It can be done using ‘manage.py’ file present at the root of the application. The python command for database migration are as follows:

In order to create migration folder

```
(venv) $ python manage.py db init
```

In order to create migration scripts and version in the migration folder

```
(venv) $ python manage.py db migrate
```

In order to apply the changes into the database without affecting the data

```
(venv) $ python manage.py db upgrade
```

All the migration scripts have been already created. In the case of change in database ‘model’, only second and third commands needs to be executed.

5.3 Flask Bootstrap

Bootstrap is an HTML, CSS and Javascript framework for developing responsive web project. If a web project contains Bootstrap integration, it can automatically adjust itself on the user requested screen, mobile tablet computer etc. It is not required to develop different front ends based on the user request. Different messages used in 'Stamp the Web' also use Bootstrap messaging. Bootstrap also helps us write less Javascript code by just calling different Bootstrap classes. In order to add a new front end module it is required to extend the 'bootstrap/wtf.html' web page in the following way:

```
{% import "bootstrap/wtf.html" as wtf %}
```

This way we can use all the Bootstrap functionalities and components [24]

5. Future System Improvements

It is attempted to make code reusable as much extent as possible, by using methods and classes. However Flask applications have a different approach. The code present inside the 'blueprints' can only be used as soon as the 'application context' is available. Some code used to send email is present in the 'main' module of the application. This code cannot be re-used for sending emails for other purposes, like automatic email sending to user, if there is a change in the timestamped content. Also if a requested page is blocked in a country of user's choice, the system sends an email to the associated user. This code executes once after 84600 seconds, and it is found that flask application context is not present and we cannot reuse this code from the 'main_blueprints'. Hence the code is re-written again in 'send_email' file. Other than this issue no other code is re-written, and it is attempted to use as little code as possible.

In case of getting text from web articles we are using simple python 'requests' module. It is working fine in case of different web articles, wikipedia and other news sources. We get the text from a user's provided URL as follows:

First we get the response from the user provided URL
```
res = requests.get(url)
```

Then convert the text present at the URL into a document.

[24] "Components · Bootstrap." 2013. 31 Jul. 2016 <http://getbootstrap.com/components/>

```
doc = Document(res.text)
```

And then we simply get document’s summary by using the appropriate method and convert this summary as an HTML document on the server, for later comparison with web pages.

```
text    =    '<!DOCTYPE    html    PUBLIC    "-//W3C//DTD    XHTML    1.0    Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1' \
      '-transitional.dtd">\n' + '<head>\n' + \
      '<meta http-equiv="Content-Type" content="text/html" ' \
      'charset="' + encoding + '">\n' + '</head>\n' + '<body>\n' \
      + '<h1>' + doc.title().split(sep='|')[0] + '</h1>'

text += doc.summary() + '</body>'
```

This approach is working fine but in some cases some text from web pages is missing. It is required to get complete text from the given URL. One possible approach could be as follows:

Rather than converting the text from a URL into a doc, we can simply get the text from response of a URL as:

```
text = res.text
```

This text also contains all the html tags and links, which needs to be removed when text is compared later with other versions. More importantly it is required to save HTML using appropriate encoding utf-8 in case windows system.

There can also other approaches to get text from HTML one is using the module beautiful soup 4 (bs4). Initially bs4 was used however extracting text using bs4 was taking more time. So the use of bs4 was removed from the code.

6 Installation

‘Stamp the Web’ requires python 3 and Flask to run. It is tested with the following flask and python packages. In order to run application smoothly, following packages are a must :

- alembic==0.8.6
- Babel==1.3
- beautifulsoup4==4.4.1
- bleach==1.4.3
- blinker==1.4
- chardet==2.3.0
- click==6.6
- cryptography==1.2.3
- cssselect==0.9.1
- docutils==0.12
- dominate==2.2.0
- enum34==1.1.2
- feedparser==5.1.3
- Flask==0.11.1
- Flask-Bootstrap==3.3.6.0
- Flask-Login==0.3.2
- Flask-Mail==0.9.1
- Flask-Migrate==1.8.0
- Flask-Moment==0.5.1
- Flask-PageDown==0.2.1
- Flask-Script==2.0.5
- Flask-SQLAlchemy==2.1
- Flask-SSLify==0.1.5
- Flask-WTF==0.12
- funcsigs==0.4
- gdata==2.0.18
- gevent==1.1.1
- greenlet==0.4.9
- gunicorn==19.4.5
- html5lib==0.9999999
- idna==2.0
- ipaddress==1.0.16
- itsdangerous==0.24
- Jinja2==2.8
- linecache2==1.0.0
- lxml==3.6.0

- Mako==1.0.4
- Markdown==2.6.6
- MarkupSafe==0.23
- mock==1.3.0
- pbr==1.8.0
- Pillow==3.1.2
- psutil==3.4.2
- psycopg2==2.6.1
- pyasn1==0.1.9
- PyChart==1.39
- pycrypto==2.6.1
- pydot==1.0.29
- Pygments==2.1
- pyinotify==0.9.6
- PyOpenGL==3.0.2
- pyOpenSSL==0.15.1
- pyparsing==2.0.3
- Pyrex==0.9.8.5
- python-dateutil==2.4.2
- python-editor==1.0
- python-ldap==2.4.22
- python-openid==2.2.5
- python-stdnum==1.2
- pytz==2014.10
- PyWebDAV==0.9.8
- PyYAML==3.11
- readability-lxml==0.6.2
- reportlab==3.3.0
- requests==2.10.0
- roman==2.0.0
- schedule==0.3.2
- simplejson==3.8.1
- six==1.10.0
- SQLAlchemy==1.0.13
- traceback2==1.4.0
- unittest2==1.1.0
- unity-lens-photos==1.0
- uTidylib==0.2
- vatnumber==1.2
- virtualenv==15.0.2
- visitor==0.1.3
- vobject==0.8.1rc0
- Werkzeug==0.11.10

- WTForms==2.1
- xlwt==0.7.5
- ZSI==2.1a1

To install a python package use the following command.

```
$ pip install [python-package]
```

A ‘requirements.txt’ file can also be used to install these packages, which is present at the root of the source code. In order to install all the packages from ‘requirements.txt’ following pip command may be used.

```
$ pip install -r requirements.txt
```

To clone this source locally into your computer.

```
$ git clone https://github.com/Sebisnow/StampTheWeb.git
```

To run the source code locally

```
$ python manage.py runserver
```

6.1 Limits for DB

Currently initial DB is created using SQlite. SQlite preserves 2 Terabytes of data. This data storage is exclusive of PDF and screenshot data storage. It is just user and their post information storage, which would be saved in a DB. We could also switch to another DB that just needs change in the configurations. Only environment variables for DB configurations needs to be changed. If configurations for another database are given, the system switches itself to that database automatically.
Currently no db username and password has not been used and variables for username and password could be added as new environment variables in the code If the other DB contains username and password.

6.2 Good Practise for Environment Variables

For better configurations it is suggested to use environment variables rather than writing them into the code. If environment variables are used and values for variables used in the code are not hardwired in the code it saves time in configurations. The setting of environment variables as shown below help us to change any value using command line and does not require to make changes in the code and update the code commit and then commit changes on the server.

```
SECRET_KEY = os.environ.get('SECRET_KEY') or 'hard to guess string'
SQLALCHEMY_COMMIT_ON_TEARDOWN = os.environ.get('SQLALCHEMY_COMMIT_ON_TEARDOWN ') or
True
MAIL_SERVER = os.environ.get('MAIL_SERVER') or 'smtp.gmail.com'
MAIL_PORT = os.environ.get('MAIL_PORT') or 587
MAIL_USE_TLS = os.environ.get('MAIL_USE_TLS') or True
MAIL_USERNAME = os.environ.get('MAIL_USERNAME') or '<username>'
MAIL_PASSWORD = os.environ.get('MAIL_PASSWORD') or '<password>'
STW_MAIL_SUBJECT_PREFIX = os.environ.get('STW_MAIL_SUBJECT_PREFIX') or '[StampTheWeb]'
STW_MAIL_SENDER = os.environ.get('STW_MAIL_SENDER') or '<stamptheweb@gmail.com>'
STW_ADMIN = os.environ.get('STW_ADMIN') or 'stamptheweb@gmail.com'
STW_POSTS_PER_PAGE = os.environ.get('STW_POSTS_PER_PAGE') or 20
STW_CHINA_PROXY = os.environ.get('STW_CHINA_PROXY') or "101.201.42.44:3128"
STW_USA_PROXY = os.environ.get('STW_USA_PROXY') or "169.50.87.252:80"
STW_UK_PROXY = os.environ.get('STW_UK_PROXY') or "89.34.97.132:8080"
STW_RUSSIA_PROXY = os.environ.get('STW_RUSSIA_PROXY') or "80.240.114.77:8000"
CHINA_PROXY = os.environ.get('CHINA_PROXY') or "60.216.40.135"
USA_PROXY = os.environ.get('USA_PROXY') or "199.115.117.212"
UK_PROXY = os.environ.get('UK_PROXY') or "90.216.222.23"
RUSSIA_PROXY = os.environ.get('RUSSIA_PROXY') or "80.240.114.77"
SERVER_URL = os.environ.get('SERVER_URL') or 'https://stamptheweb.org'
SQLALCHEMY_TRACK_MODIFICATIONS = os.environ.get('SQLALCHEMY_TRACK_MODIFICATIONS') or
False
```

6.2.1 Description for Environment Variables

Name	Description
`SECRET_KEY`	The key for generating login session. By default, user sessions are stored in client-side cookies that are cryptographically signed using the configured SECRET_KEY. Any tampering with the cookie content would render the signature invalid, thus invalidating the session. [25]
`SQLALCHEMY_COMMIT_ON_TEARDOWN`	Set true if automatic commits of DB are

[25] Grinberg, Miguel. *Flask Web Development: Developing Web Applications with Python*. " O'Reilly Media, Inc." 2014.

	required using SQLAlchemy
`MAIL_SERVER`	The mail server email which requires to send email to the user for different purposes.
`MAIL_PORT`	The port for SSL or TLS which is being used by the mail server.
`MAIL_USE_TLS`	‘True’ if Mail server uses TLS protocol.
`MAIL_USERNAME`	Username of email account used to send emails to the users.
`MAIL_PASSWORD`	Password of email account used to send emails to the users.
`STW_MAIL_SUBJECT_PREFIX`	The subject prefix of each mail send by the system.
`STW_MAIL_SENDER`	Mail sender email address of the system.
`STW_ADMIN`	Admin email address of the system.
`STW_POSTS_PER_PAGE`	How many posts needs to be displayed on a single page?
`STW_CHINA_PROXY`	A valid public IP address of a proxy of china. To load a webpage from China.
`STW_USA_PROXY`	A valid public IP address of a proxy of USA. To load a webpage from USA.
`STW_UK_PROXY`	A valid public IP address of a proxy of UK. To load a webpage from UK.
`STW_RUSSIA_PROXY`	A valid public IP address of a proxy of Russia. To load a webpage from Russia.
`CHINA_PROXY`	Public IP of the China proxy without the port number. This is to get the location of the proxy.
`USA_PROXY`	Public IP of the USA proxy without the port number. This is to get the location of the proxy.
`UK_PROXY`	Public IP of the UK proxy without the port number. This is to get the location of the proxy.

`RUSSIA_PROXY`	Public IP of the Russia proxy without the port number. This is to get the location of the proxy.
`SERVER_URL`	The URL of the server currently used. In order to send scheduled emails to the user. For e.g 'https://stamptheweb.org'
`SQLALCHEMY_TRACK_MODIFICATIONS`	Set 'True' if system should generate URL automatically for sub domains. However in our case some problems found, so it is set 'False'

6.3 Reporting Bug section for source code

For reporting bugs and issues in the system a github issue tracker has been set. It is very important for working in teams and collaboratively solving issues. Github provides a sophisticated issue tracking system associated with the source code. It can be accessed using.

Stamp The Web Issue tracking: URL

6.4 Future Improvements

6.4.1 Articles by modifications

The system may include statistics for articles which have highest modification once timestamped in the system. Then the user could discover which domains/ media outlets tend to modify their reporting, maybe also their political opinion.

6.4.2 Help

Help and reporting bugs sections are not included in the system. It is necessary that user is able to find some components in the system. Help section may also contain videos that are performing certain tasks in the system, this can also make system very helpful.

6.4.3 Consistent Hash

It is noticed with some web articles that their hash is not consistent even if the content of the web article is not changed. When a web article is compared with another version, even if there is no change in the web article system finds a changed hash, hence a flash message appear ‘Change in content found’. This needs to be fixed.

6.4.4 Search for Blocked Articles

Currently search works for all the timestamped articles. We can simply browse blocked web articles by going to ‘where it is blocked’ page. We also need to add search option for blocked articles.

6.4.5 Check Multiple proxies

In order to find if a web page is blocked in a country, we only check one proxy. If that proxy is unable to find the requested page. It is shown blocked in that country. Sometimes a proxy is not working and in that case we get the result as the page is blocked. We need to add more than one proxy at least 3 for a country. In order to add more proxies the current code needs to be replaced with the following:

```
proxy1 = '169.50.87.252:80'
proxy2 = '89.34.97.132:8080'
proxy3 = '11.22.33.44:80'
requests.get(url, proxies={"http":proxy1,"http":proxy2 ,"http":proxy3})
```

Document Control

Title:	Stamp the Web Developer’s Documentation
Issue:	Issue 1
Date:	10 August 2016
Author:	Waqar Detho
Filename:	technical_doc

Document Signoff

Nature of Signoff	**Person**	**Signature**	**Date**	**Role**
Authors	Waqar Detho			Software Developer

Reviewers				

Document Change Record

Date	**Version**	**Author**	**Change Details**
10 August 2016	Issue 1 Draft 2	Waqar Detho	First complete draft