

Final Presentation
Summer 2017

Tidbit: Automated Document Summarization 1.0

Team Members: Alastair Paragas & Alberto Mizrahi

Product Owner: Nagarajan Prabakar

Instructor: Masoud Sadjadi

School of Computing and Information Sciences
Florida International University



Project definition

Problem:

- During the course of their day, professionals have to constantly read articles and documents to keep their knowledge up to date.
- This process requires a great deal of effort as well as a significant time investment.

Solution:

- An application that would summarize articles and documents would save a tremendous amount of time and energy.
- For ease of use and efficiency, the application should allow the user to provide a URL that points to the article.
- The system should then present the user with a concise summary of the most important ideas of the document.

Project definition

🔍 Tidbit

✕ LOGOUT

Enter a url of a site you would like to have summarized

<https://www.bloomberg.com/news/articles/2017-06-25/takata-seeks-u-s-bankruptcy-protection-after-air-bag-recalls>

Luhn

🔍 SUMMARIZE

Success

Algorithm

Summarization algorithm used

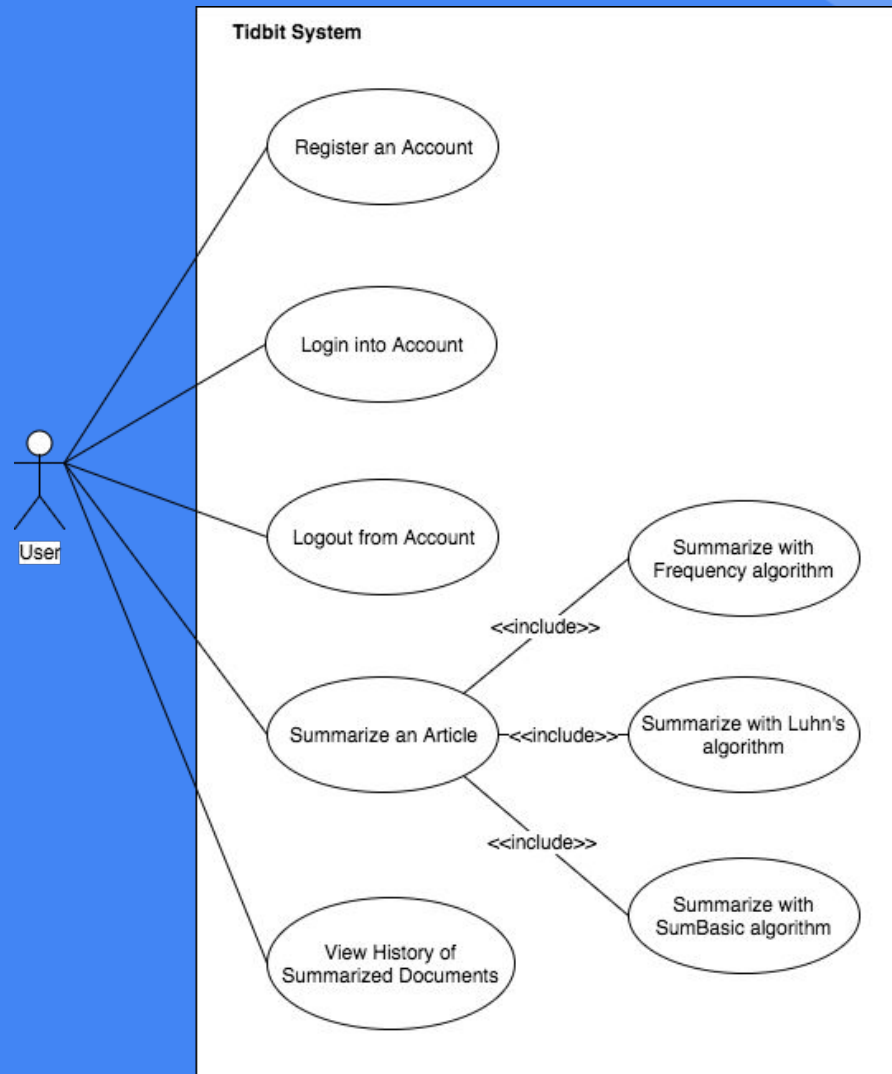
LUHN

Summary

Summary obtained

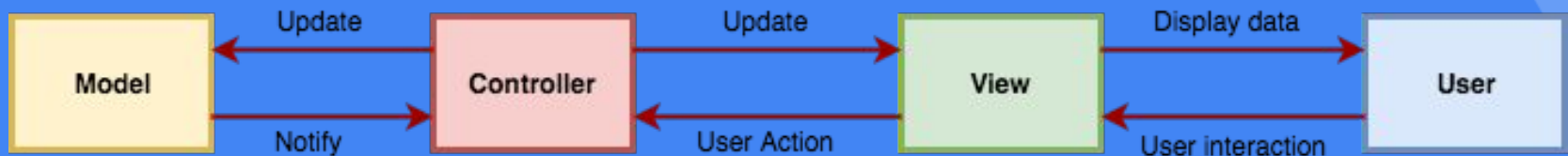
Takata Corp. filed for bankruptcy protection in the biggest postwar Japanese corporate failure in the manufacturing industry as the 84-year-old company buckled under liabilities from millions of recalled air bags and agreed to be sold to a Chinese-backed investor. The Chapter 11 bankruptcy in Delaware listed more than \$10 billion in liabilities, including claims from automakers including Honda Motor Co. – the biggest user of the air bags – and Toyota Motor Corp. as well as individuals who have brought class-action lawsuits. “This will be a long process under the best of circumstances, and Takata going bankrupt, though not surprising, only adds to a potential increase in the time it takes to replace tens of millions of airbags,” Brauer, executive publisher of Kelley Blue Book, said Monday in an emailed statement. “We do not expect a Takata bankruptcy to have an impact on claims pending against auto manufacturer defendants for their role in the airbag scandal,” Peter Prieto, court-appointed chair lead counsel for the consumer plaintiffs in the Takata Airbag Product Liability Litigation, said in an emailed statement. In the U.S. alone, about 43 million air bag inflators are currently subject to recall, and only about 38 percent have been repaired as of May 26, according to data on the U.S. Department of Transportation’s National Highway Traffic Safety Administration’s website.

Requirements: Use Cases



System Design: Architecture

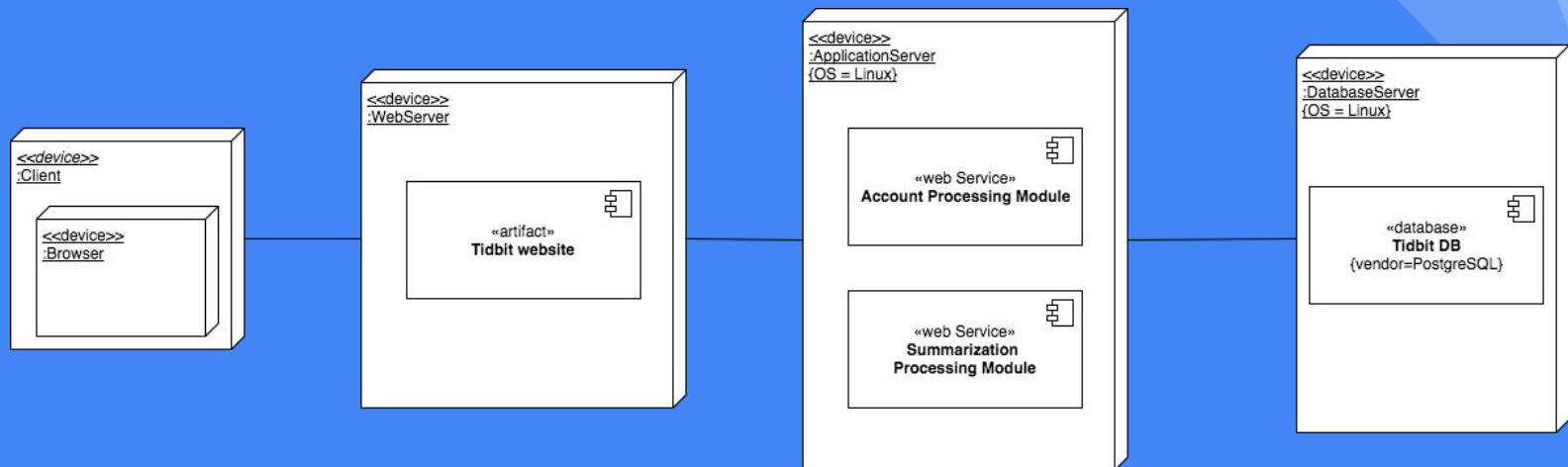
Model-View-Controller (MVC) Architecture:



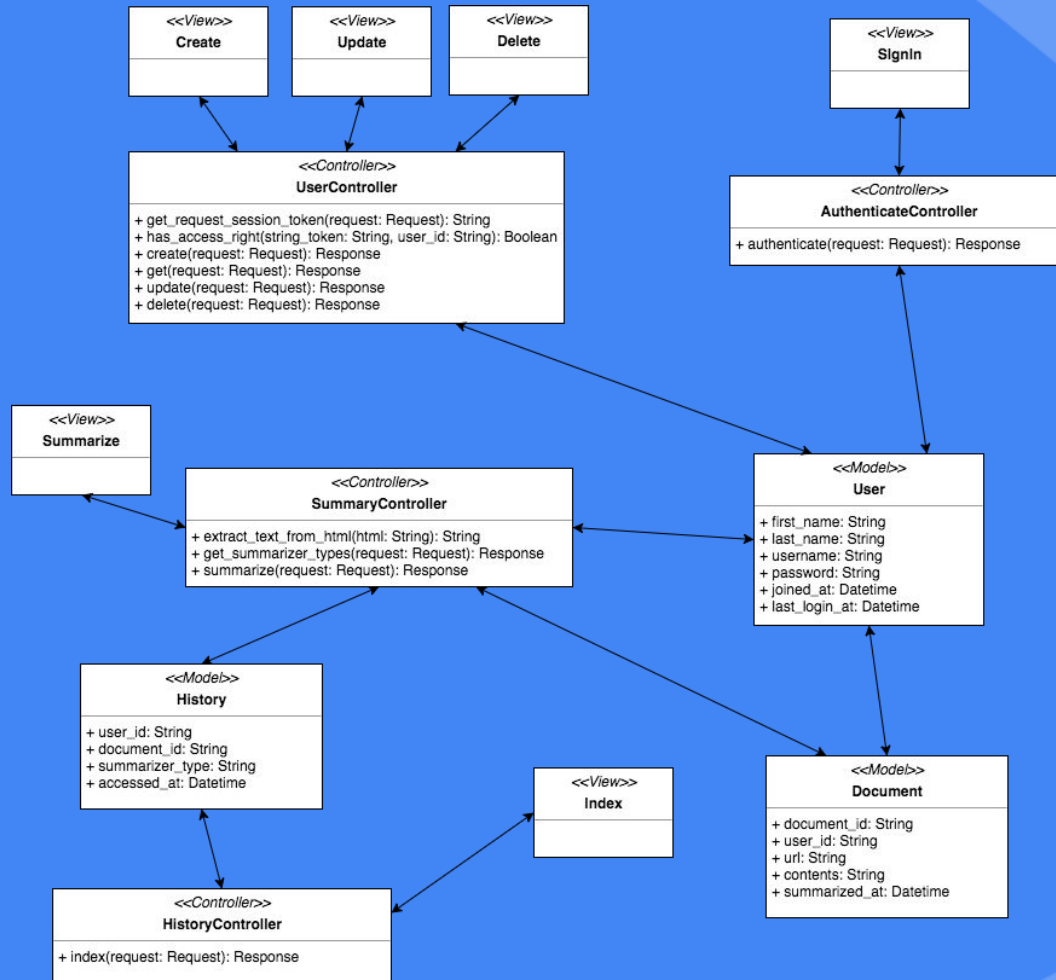
System Design: Requirements

- **Hardware:** Any computer running Linux, Mac OS or Windows.
- **Software:** Python v3.6, Flask v0.12, Docker v17.06.0-ce, PostgreSQL v9.6.3, Pip v3, Git 2.13.3, GitHub, React.js 15.0, Highland.js v3, Bootstrap v3
 - **Python libraries:** Aiohttp v2.1.0, Aiopg v0.13.0, Aitertools v0.1.0, Aniso8601 v1.2.0, Async-timeout v1.2.1, Asynctest v0.10.0, Cerberus v1.1, Chardet v3.0.3, Click v6.7, Flask-RESTful v0.3.5, Itsdangerous v0.24, Jinja2 v2.9.5, MarkupSafe v1.0, Multidict v2.1.6, Neo4j-driver v1.1.0, Neomodel v3.2.2, Psycopg2 v2.7.1, PyJWT v1.5.0, Python-dateutil v2.6.0, Pytz v2016.10, Six v1.10.0, Werkzeug v0.12.1, Yarl v0.10.2, BeautifulSoup4 v4.6.0, Readability-lxml v0.6.2, Six v1.10.0, Typing v3.6.1, Werkzeug v0.12.1, Yarl v0.10.2

System Design: Deployment



Minimal Class Diagram



Design Patterns

Throughout the course of the system's development, a variety of software design patterns were utilized so as to ensure that our code is efficient, reusable and flexible. The following are some examples that can be found in our codebase:

- Object pool
- Lazy initialization
- Front controller
- Modules
- Command
- Iterator
- Strategy
- Event-based asynchronous

User Stories

1. Implement Luhn's algorithm and the SumBasic algorithm for text summarization.
2. Set up the basic back-end API
3. Design the front end and deploy the website
4. Implement the Summarization API routes
5. Understand the use of Streams with Haskell and Conduit-Haskell
6. Implement the frequency summarization algorithm.

User Story #1

Description: As a User, I would like that the application allow me to choose from various summarization algorithms when summarizing a document so I can see which one works best for me.

Acceptance Criteria:

- The team should implement another summarization algorithm besides the frequency one they have already implemented.
- The team must have tested this algorithm as thoroughly as possible.

User Story #1 (Cont.)

SumBasic (Nenkova and Vanderwende, 2005) is a system that produces generic multi-document summaries. Its design is motivated by the observation that words occurring frequently in the document cluster occur with higher probability in the human summaries than words occurring less frequently. Specifically, SumBasic uses the following algorithm:

Step 1 Compute the probability distribution over the words w_i appearing in the input, $p(w_i)$ for every i ; $p(w_i) = \frac{n}{N}$, where n is the number of times the word appeared in the input, and N is the total number of content word tokens in the input.

Step 2 For each sentence S_j in the input, assign a weight equal to the average probability of the words in the sentence, i.e.,

$$Weight(S_j) = \sum_{w_i \in S_j} \frac{p(w_i)}{|S_j|}$$

Step 3 Pick the best scoring sentence that contains the highest probability word.

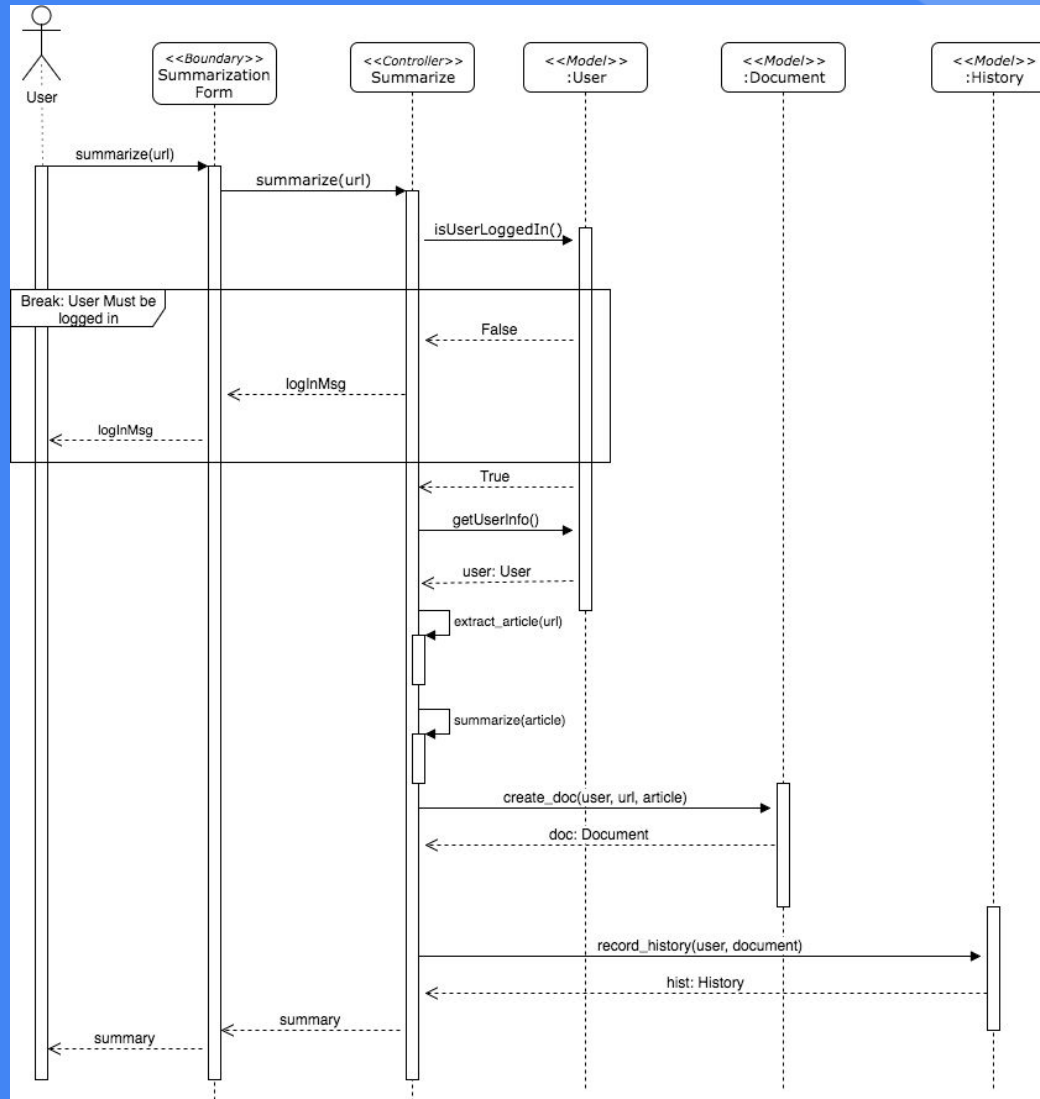
Step 4 For each word w_i in the sentence chosen at step 3, update their probability:

$$p_{new}(w_i) = p_{old}(w_i) \cdot p_{old}(w_i)$$

Step 5 If the desired summary length has not been reached, go back to Step 2

Reference: Vanderwende, Lucy, Hisami Suzuki, Chris Brockett, and Ani Nenkova. "Beyond SumBasic: Task-focused Summarization with Sentence Simplification and Lexical Expansion." Information Processing & Management 43.6 (2007): 1606-618. Web.

User Story #1 (Cont.)



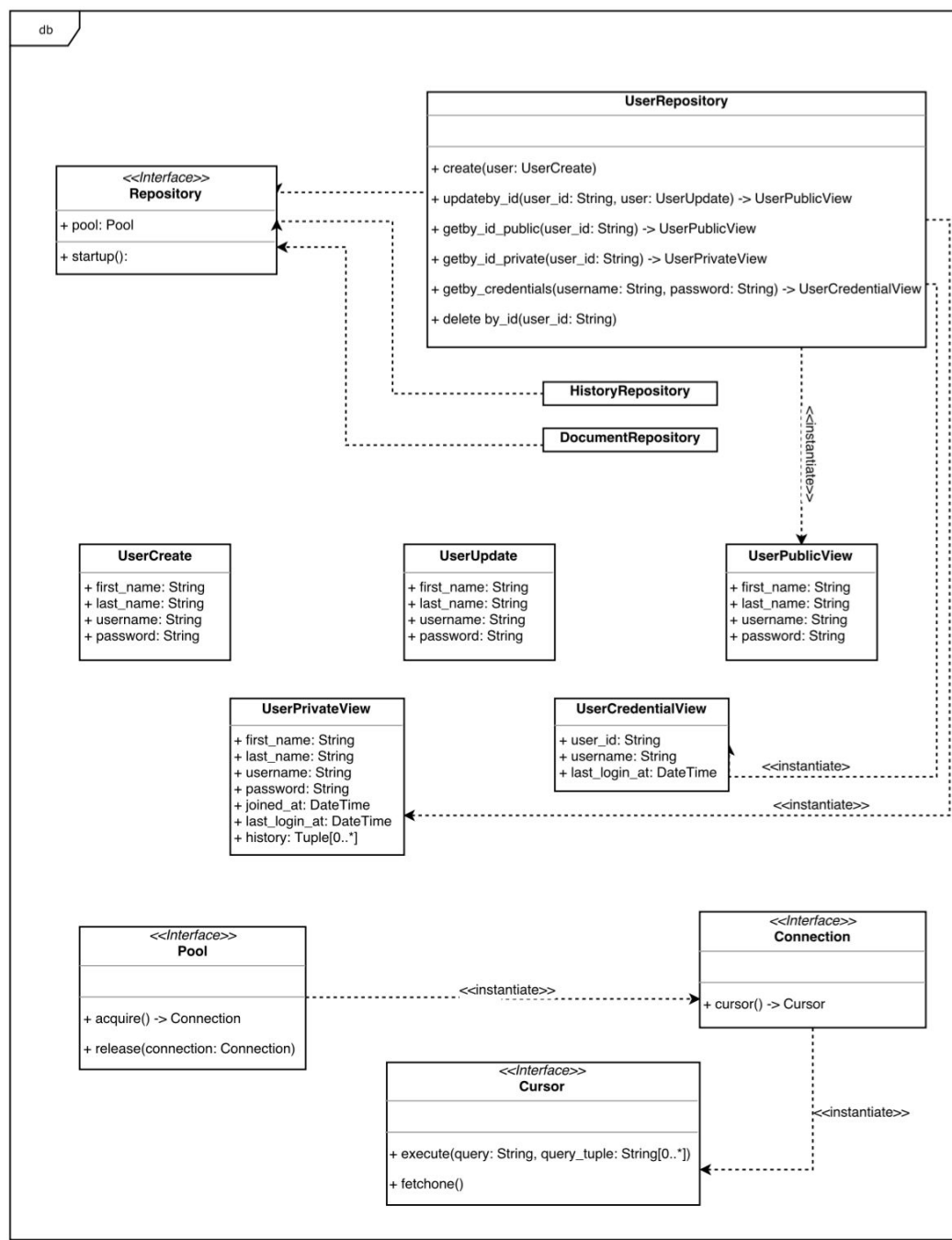
User Story #2

Description:

- As a Researcher, I would like for the team to develop the basic structure of the back-end API (which includes support for user login, logout, etc.) so that the application's foundation will be working correctly.

Acceptance Criteria:

- The team must have chosen and set up the libraries and frameworks that they will use to develop the back-end.
- The team must have added support in the API for the User resource, with the ability to create, delete, get and update users in the system.
- The team must have created documentation explaining the functionality added to the API



User Story #2

User Story #3

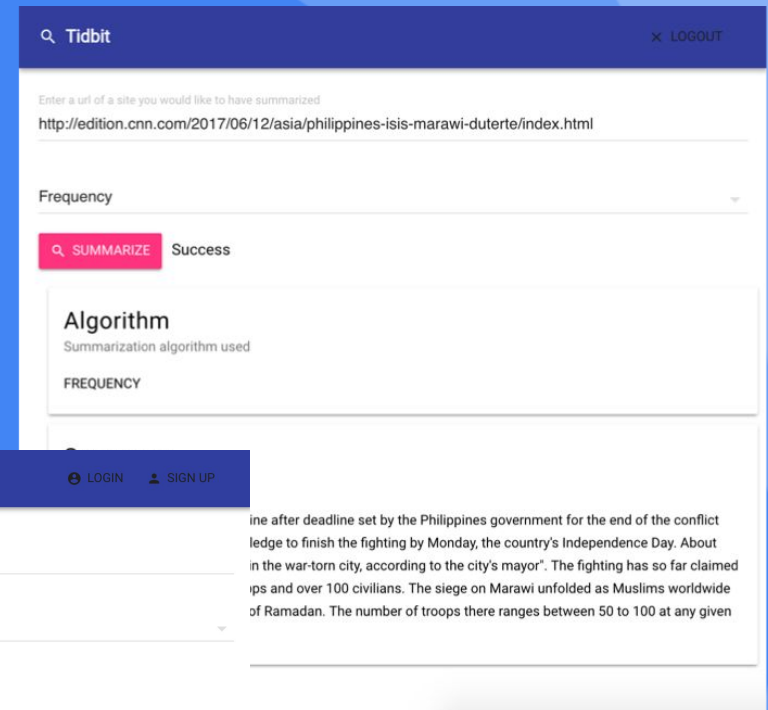
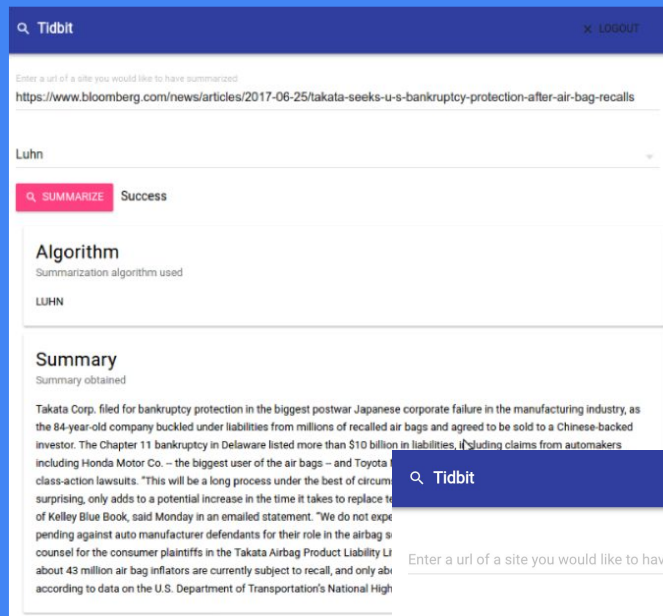
Description:

- As a User, I would like that the UI of the application should be designed as a professional web app so that it is easy as well as enjoyable to use.
- As a User, I would like for the application to be available online so that I can access it wherever I go.

Acceptance Criteria:

- The team must have developed a front end UI for the application.
- The front end must be hooked up to the back end API, and hence the application should be working correctly.
- The website must be deployed to a live production server.

User Story #3



User Story #4

Description:

- As a User, I would like for the application to keep track of the documents I have summarized in the past so that I can check this history any time I wish.
- As a Data Analyst, I would like for the team to keep track of the documents that the system has summarized so far so that we can conduct statistical tests on this data to obtain insights on our users activities and preferences.

Acceptance Criteria:

- Implemented summarization API routes for documents and history resources.
- Unit tested summarization API routes for documents and history resources.

User Story #4 (Cont.)

Tidbit

LOGOUT

Enter a url of a site you would like to have summarized

<https://www.bloomberg.com/news/articles/2017-06-25/takata-seeks-u-s-bankruptcy-protection-after-air-bag-recalls>

Luhn

SUMMARIZE

Success

Algorithm

Summarization algorithm used

LUHN

Summary

Summary obtained

Takata Corp. filed for bankruptcy protection in the biggest postwar Japanese corporate failure in the manufacturing industry, as the 84-year-old company buckled under liabilities from millions of recalled air bags and agreed to be sold to a Chinese-backed investor. The Chapter 11 bankruptcy in Delaware listed more than \$10 billion in liabilities, including claims from automakers including Honda Motor Co. – the biggest user of the air bags – and Toyota Motor Corp. as well as individuals who have brought class-action lawsuits. "This will be a long process under the best of circumstances, and Takata going bankrupt, though not surprising, only adds to a potential increase in the time it takes to replace tens of millions of airbags," Brauer, executive publisher of Kelley Blue Book, said Monday in an emailed statement. "We do not expect a Takata bankruptcy to have an impact on claims pending against auto manufacturer defendants for their role in the airbag scandal," Peter Prieto, court-appointed chair lead counsel for the consumer plaintiffs in the Takata Airbag Product Liability Litigation, said in an emailed statement. In the U.S. alone, about 43 million air bag inflators are currently subject to recall, and only about 38 percent have been repaired as of May 26, according to data on the U.S. Department of Transportation's National Highway Traffic Safety Administration's website.

User Story #5

Description:

- As a Programmer, I need to be confident with using Haskell and streaming libraries like Conduit so that we understand how they work given that we will be using Haskell as one of the two main programming languages in this project and we are potentially going to be creating streams of data flow into the system.
- As a Researcher, I would like to be able to see streaming Twitter tweets on a console because we want to verify that a system could be built that allows for streaming data.

Acceptance Criteria:

- Have a sample Twitter feed implementation using the Conduit Library and Haskell

User Story #5

```
1 module Main where
2
3 import Control.Monad.IO.Class (liftIO)
4 import Control.Monad.Trans.Resource (runResourceT)
5 import Data.Conduit (($+~), ($+=), runConduit)
6 import Data.Conduit.List (mapM_, map, filter, catMaybes)
7 import qualified Data.ByteString.Char8 (pack)
8 import Data.Text (unpack)
9 import qualified Data.Text (pack)
10 import Data.Maybe (fromJust)
11 import Configuration.Dotenv (loadFile, onMissingFile)
12 import System.Environment (lookupEnv)
13 import System.Exit (die)
14 import Web.Authenticate.OAuth (oauthConsumerKey, oauthConsumerSecret)
15 import Web.Twitter.Conduit (Credential(Credential), newManager, tlsManagerSettings)
16 import Web.Twitter.Types
17   (StreamingAPI(SStatus, SRetweetedStatus)
18   , Status(Status), statusText, statusLang
19   , RetweetedStatus(RetweetedStatus), rsRetweetedStatus
20   )
21 import Web.Twitter.Conduit.Stream (statusesFilterByTrack, stream)
22 import Web.Twitter.Conduit.Types
23   (TWInfo(TWInfo), TWToken(TWToken)
24   , twitterOAuth, twToken, twOAuth, twCredential, twProxy
25   )
26
27
28 filterEnglishTweets :: StreamingAPI -> Bool
29 filterEnglishTweets tweet =
30   let
31     langIsEnglish (Status {statusLang=language}) = case language of
32       Just "en" -> True
33       _ -> False
34   in case tweet of
35     SStatus statusObj -> langIsEnglish statusObj
36     SRetweetedStatus (RetweetedStatus {rsRetweetedStatus=statusObj}) ->
37       langIsEnglish statusObj
38     _ -> False
39
40
41 tweetParser :: StreamingAPI -> Maybe String
42 tweetParser tweet = case tweet of
43   SStatus (Status {statusText=status}) -> Just $ unpack status
44   SRetweetedStatus (RetweetedStatus {rsRetweetedStatus=rstatus}) ->
45     Just $ unpack $ statusText rstatus
46   _ -> Nothing
```

```
49 main :: IO ()
50 main = do
51
52   let
53     loadFileIO =
54       loadFile False "./.env"
55     errorHandlerIO =
56       putStrLn ".env file is missing! Using global env instead!"
57     in onMissingFile loadFileIO errorHandlerIO
58
59   consumerKey <- lookupEnv "TWITTER_CONSUMER_KEY"
60   consumerSecret <- lookupEnv "TWITTER_CONSUMER_SECRET"
61   accessToken <- lookupEnv "TWITTER_ACCESS_TOKEN"
62   accessTokenSecret <- lookupEnv "TWITTER_ACCESS_TOKEN_SECRET"
63   case (consumerKey, consumerSecret, accessToken, accessTokenSecret) of
64     (Just _, Just _, Just _, Just _) -> return ()
65     (_, _, _, _) -> die "required Environment Variables not set!"
66
67   connectionManager <- newManager tlsManagerSettings
68
69   let
70     apiRequest = statusesFilterByTrack $ Data.Text.pack "*"
71     twitterInfo = TWInfo {
72       twToken = TWToken {
73         twOAuth = twitterOAuth {
74           oauthConsumerKey = (Data.ByteString.Char8.pack . fromJust) consumerKey,
75           oauthConsumerSecret = (Data.ByteString.Char8.pack . fromJust) consumerSecret
76         },
77         twCredential = Credential [
78           (Data.ByteString.Char8.pack "oauth_token",
79            (Data.ByteString.Char8.pack . fromJust) accessToken
80           ),
81           (Data.ByteString.Char8.pack "oauth_token_secret",
82            (Data.ByteString.Char8.pack . fromJust) accessTokenSecret
83           )
84         ]
85       }
86     },
87     twProxy = Nothing
88   }
89
90   in runResourceT $ do
91     stream <- stream twitterInfo connectionManager apiRequest
92     stream $+=
93       Data.Conduit.List.filter filterEnglishTweets $+=
94       Data.Conduit.List.map tweetParser $+=
95       Data.Conduit.List.catMaybes $+=
96       Data.Conduit.List.mapM_ (liftIO . putStrLn)
```

User Story #6

Description:

- As as User, I would like to be able to summarize an article so that it saves me the time to have to read through its entirety.

Acceptance Criteria:

- The team must have a working version of a summarization algorithm.
- The team must have tested this algorithm as thoroughly as possible.

User Story #6 (Cont.)

Tidbit

×

LOGOUT

Enter a url of a site you would like to have summarized

http://edition.cnn.com/2017/06/12/asia/philippines-isis-marawi-duterte/index.html

Frequency

Q SUMMARIZE

Success

Algorithm

Summarization algorithm used

FREQUENCY

Summary

Summary obtained

As the fighting wears on, deadline after deadline set by the Philippines government for the end of the conflict has been missed, including a pledge to finish the fighting by Monday, the country's Independence Day. About 1,000 civilians remain trapped in the war-torn city, according to the city's mayor". The fighting has so far claimed the lives of 58 government troops and over 100 civilians. The siege on Marawi unfolded as Muslims worldwide began to mark the holy month of Ramadan. The number of troops there ranges between 50 to 100 at any given time, the Pentagon said.

Test Suites and Test Cases

- Python's default unittest library was used to test the backend.
- To test the asynchronous server I/O code, we used the async-test library.
- Selenium IDE was used to test the frontend.
- Unit tests were written after every user story where code was implemented.
- After every user story, integration tests were run to ensure the correct functioning of the system

Test Suite: Examples

- **Test case ID:** UserRepository_test_getpublic_exists
- **Description/Summary of Test:** Tests that when the public information of a User is requested (which includes his or her first name, last name and username), the information is provided.
- **Pre-condition:** The database must already contain an existing User.
- **Expected Results:** The appropriate information is returned.
- **Actual Result:** The appropriate information is returned.
- **Status (Fail/Pass):** Pass
- **Testing tools:** unittest, AsyncTest

Test Suite: Examples

- **Test case ID:** UserRepository_test_getbycredentials_exists
- **Description/Summary of Test:** Tests that when the credential informations of a User is requested (which includes his or her user ID and username), the information is returned as long as the entity requesting this information is authorized to see it.
- **Pre-condition:** The database must already contain an existing User.
- **Expected Results:** The User's credential information is returned as long as the entity requesting it is authorized to do so.
- **Actual Result:** The User's credential information is returned as long as the entity requesting it is authorized to do so.
- **Status (Fail/Pass):** Pass
- **Testing tools:** Unittest, Asyncctest

Summary

- **Tidbit** is a web application that utilizes advanced NLP algorithms to summarize documents and articles.

Tidbit saves its users' time and energy by allowing them to read more, quicker.

- Contact information:

- Alastair Paragas: alastairparagas@gmail.com

- Alberto Mizrahi: albertomizrahib@gmail.com

- Questions?

- Thank You!



Bootstrap



PostgreSQL



React

