|  |  |
| --- | --- |
| PROJECT DESIGN  Week 4 Assignment | ABSTRACT  This document was created for UMUC Course, CMSC 495, and analyzes aspects of the (TNC)  Group 3 Members  Name: Christiano, Andrew  Name: Fernandez, Yrume  Name: Orwick, Brian  Name: Sell, Julia  Class: CMSC 495 - Current Trends and Projects in Computer Science Professor: Dr. Hung Dao  Due: 16 September 2018 |

**Version Control**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Revision # | Date | Name | Descriptions | Contact Info |
| TNC\_0001 | 9/11/2018 | Yrume Fernandez | Created | [Yrume.fernandez@gmail.com](mailto:Yrume.fernandez@gmail.com) |
| TNC\_0002 | 9/13/2018 | Yrume Fernandez | Revisions | [Yrume.fernandez@gmail.com](mailto:Yrume.fernandez@gmail.com) |

Table of Contents

[1. The News Code (TNC) Class Diagram 3](#_Toc524638802)

[2. Event-Trace Diagram 3](#_Toc524638803)

[3. Class Design 4](#_Toc524638804)

[4. Unresolved Risks and Risk Mitigations: 6](#_Toc524638805)

# The News Code (TNC) Class Diagram

# Event-Trace Diagram

Scenario 1:

A user opens up a web-browser on their computer system and requests a website by inputting the Uniform Resource Locator (URL) for the Trusted News Code (TNC) front-end services. This intern initiates a request from the front-end service to the back-end service which sets the conditions (creating required variables, implementing methods) to retrieve news articles from hard-coded news sites. These news sites return specific information required which is then parsed, and stored in a database. Finally, the information is pulled from the database by the front-end services and displayed to the user in html format.



Figure - Event Trace Diagram

# Class Design

1. Front-End Subsystem:
   1. # Provide user with a web-based user interface and Initiate Trusted News Code
   2. # Request information from SQLite Database for web-based user interface display
2. Back-End Subsystem:
   1. # import required modules for web scraping, html parsing, multi-threading and database
      1. import requests
      2. import newspaper
      3. from newspaper import news\_pool
      4. import sqlite3
   2. # create a list containing news sites to scrape
      1. web\_list = ['http://www.foxnews.com', 'http://www.usatoday.com', 'http://www.cnn.com']
   3. # setup newspaper instance to multi-thread news sources
      1. newsWebList = [newspaper.build(i) for i in web\_list]
      2. news\_pool.set(newsWebList, threads\_per\_source=2)
      3. news\_pool.join()
   4. # connect to SQLite database and initiate / build table
      1. con = sqlite3.connect('tnc.db')
      2. with con:
      3. cur = con.cursor()
      4. cur.execute("DROP TABLE IF EXISTS NewsArticle")
      5. cur.execute("CREATE TABLE NewsArticle(Id TEXT, Number INT, Name TEXT, Count INT)")
   5. # Define The News Counter Webscraper and Source Interator
      1. def tncWebscraper():
      2. # iterates through sources
      3. for web\_page in web\_list:
      4. # set get request for html
      5. i = 0
      6. j = 1
      7. for article in newsWebList[j].articles:
      8. print (i)
      9. article.download()
      10. article.parse()
      11. paragraphs = article.title
      12. #print(web\_page, " ", paragraphs)
      13. cur.execute("INSERT INTO NewsArticle VALUES(?,?,?,?)", (web\_page, i, paragraphs, 1))
      14. i = i+1
      15. j = j + 1
   6. # Retrieves Database Information for Front-End Services
      1. def dbRetrieve():
      2. cur.execute("SELECT \* FROM NewsArticle")
      3. data=cur.fetchall()
      4. for line in data:
      5. print(line)
   7. # Compares articles one by one for commonality and trustworthiness
      1. def compareArticle():
      2. wordcount = 0
      3. totalcount= 0
      4. for web\_page in web\_list:
      5. cur.execute("SELECT \* FROM NewsArticle WHERE Id = ?;", (web\_page,))
      6. words1=cur.fetchall()
      7. cur.execute("SELECT \* FROM NewsArticle WHERE Id != ?;", (web\_page,))
      8. words2=cur.fetchall()
      9. for line in words1:
      10. site, id, title, count = line
      11. word = title.split()
      12. for line2 in words2:
      13. site1, id1, title1, count1 = line2
      14. words = title1.split()
      15. for x in word:
      16. totalcount= totalcount + 1
      17. for y in words:
      18. if x == y:
      19. wordcount = wordcount + 1
      20. if wordcount / totalcount \* 100 >= 70:
      21. print("will be sent to database count")
      22. #for testing
      23. print("total: ", totalcount, "word: ", wordcount)
      24. print(line)
      25. print(line2)
      26. else:
      27. print("Did not make it")
      28. print("total: ", totalcount, "word: ", wordcount)
      29. totalcount= 0
      30. wordcount= 0

# Unresolved Risks and Risk Mitigations:

Potential Risk and Mitigation

Potential risk when operating the system are:

1. The system may fail to retrieve information from trusted websites (due to https certificate error, no internet connection, or other unknown circumstances)
   1. Mitigation: System will check for error code 200 to ensure website is available prior to submitting get-request for html.
2. System may fail to provide adequate information if fake-news goes viral and a number of articles are found on the internet based on false information.
   1. Mitigation: System will have a fake-news option that will lower the trustworthiness of the article.