SMD Project 3

Part 1 - Software Design

11 October , 2015

Group Number: 13

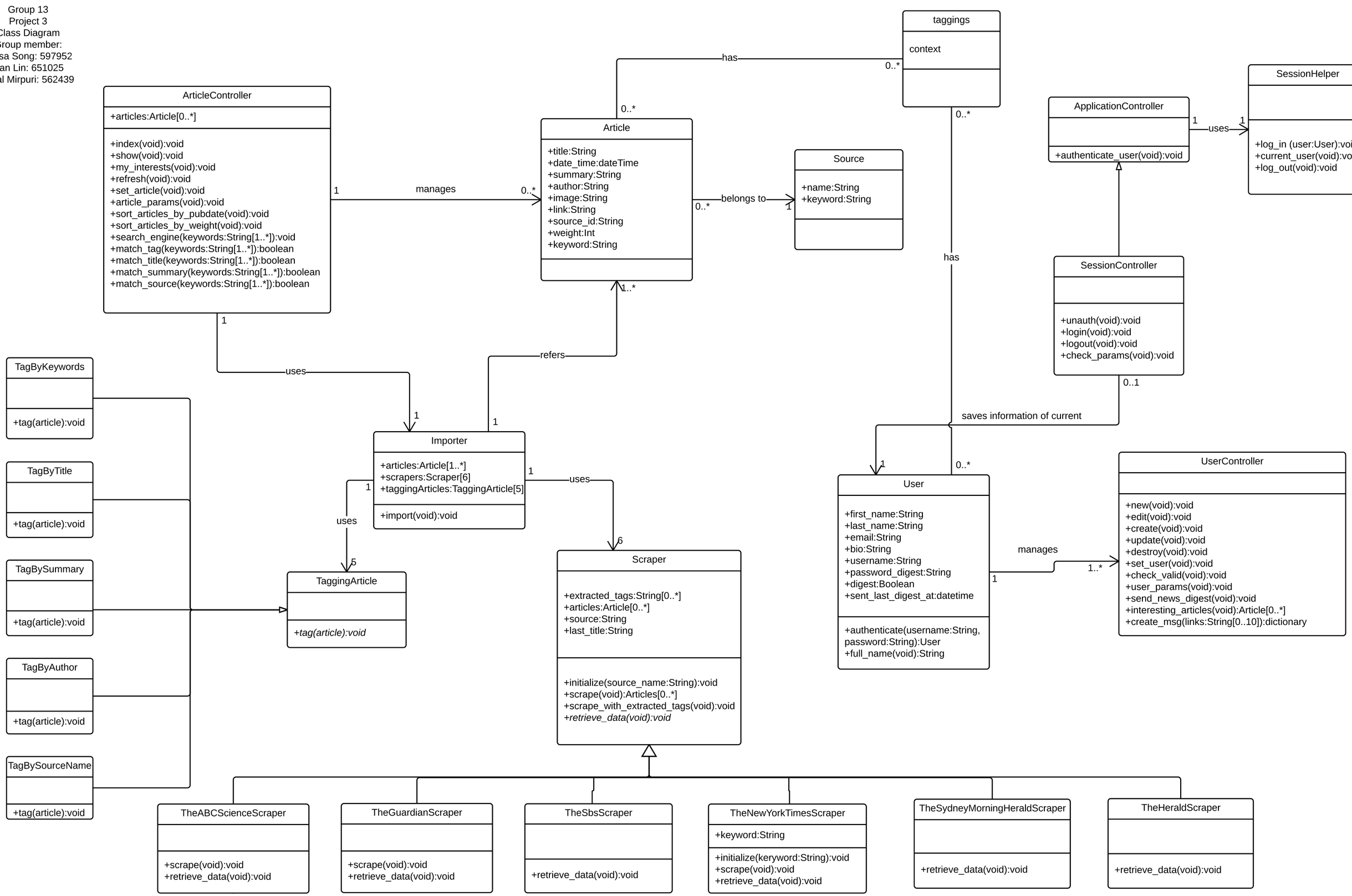
Group Member:

Nan Lin: 651025

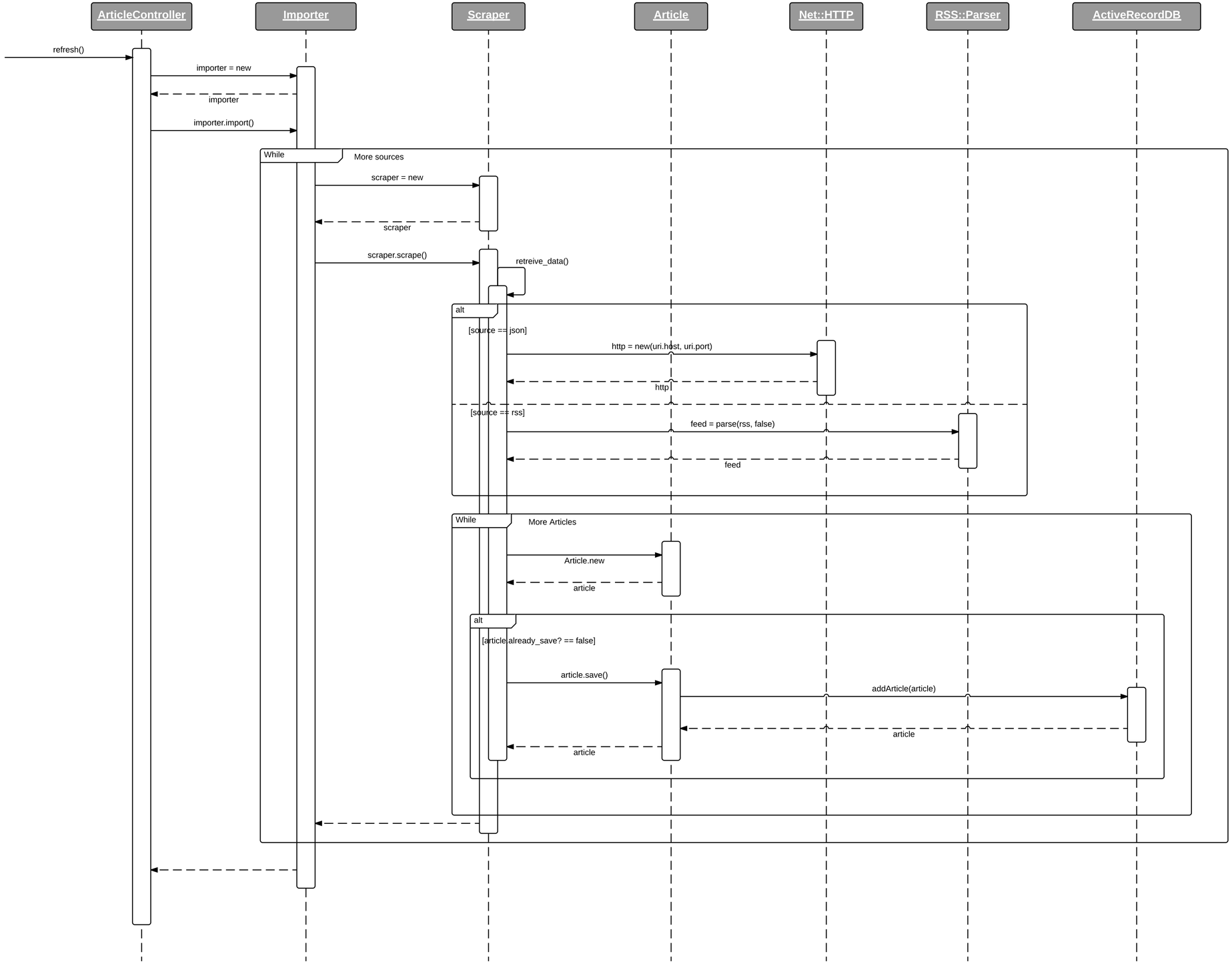
Tessa Song: 597952

Nihal Mirpuri: 562439

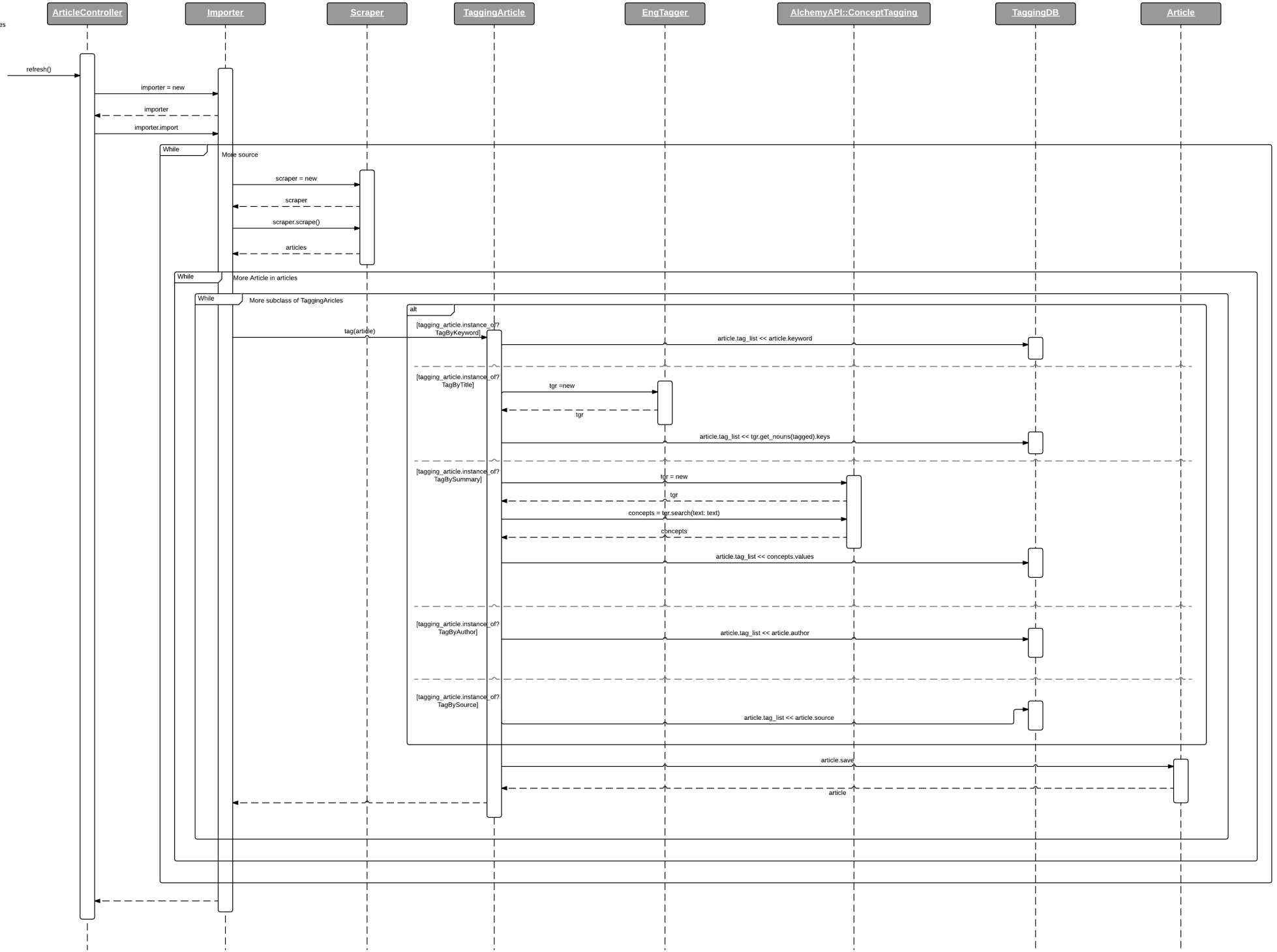
Group 13
Project 3
Class Diagram
Group member:
Tessa Song: 597952
Nan Lin: 651025
Nihal Mirpuri: 562439



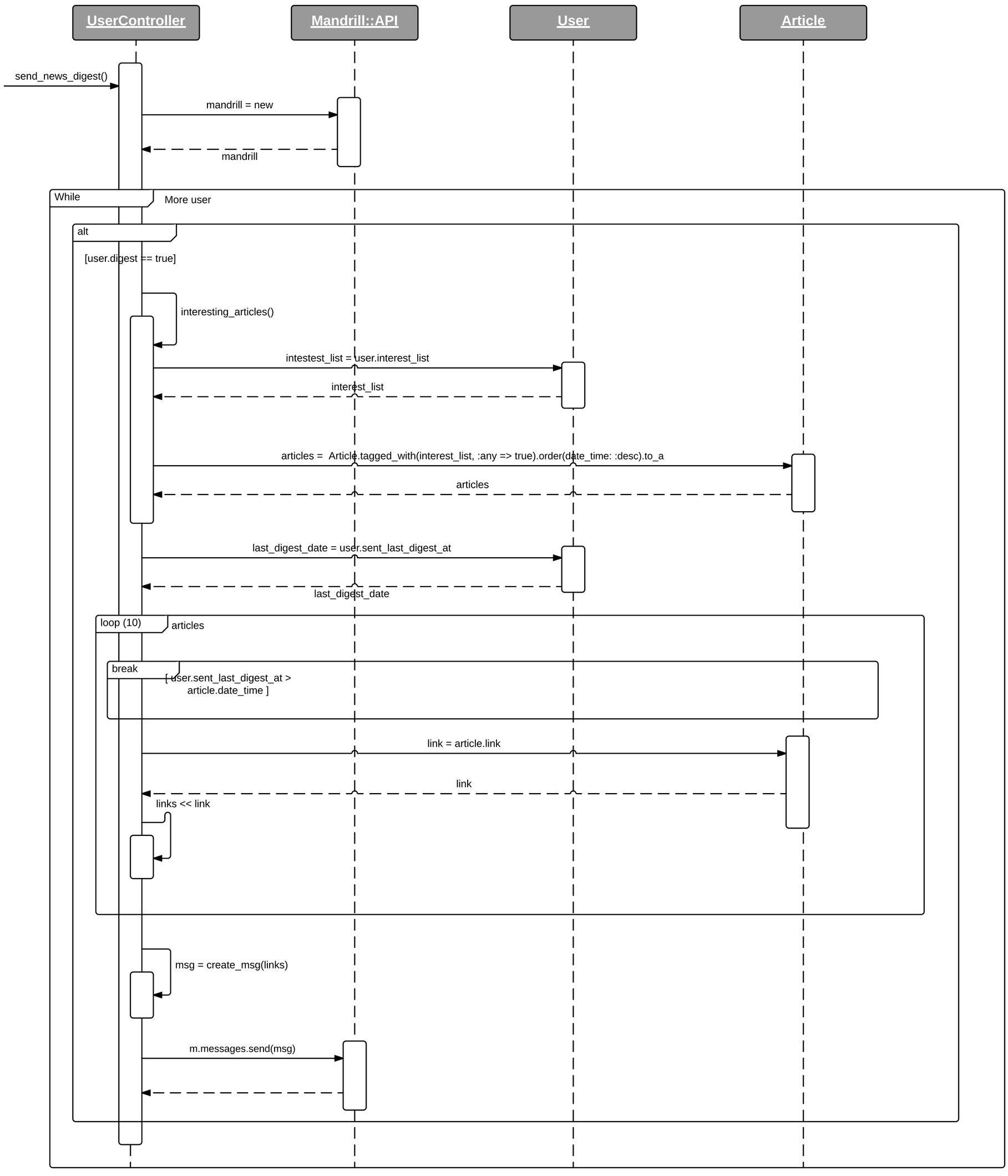
Group 13
Project 3 Class Diagram
Group member:
Nihal Mirpuri: 562439
Nan Lin: 651025
Tessa Song: 597952



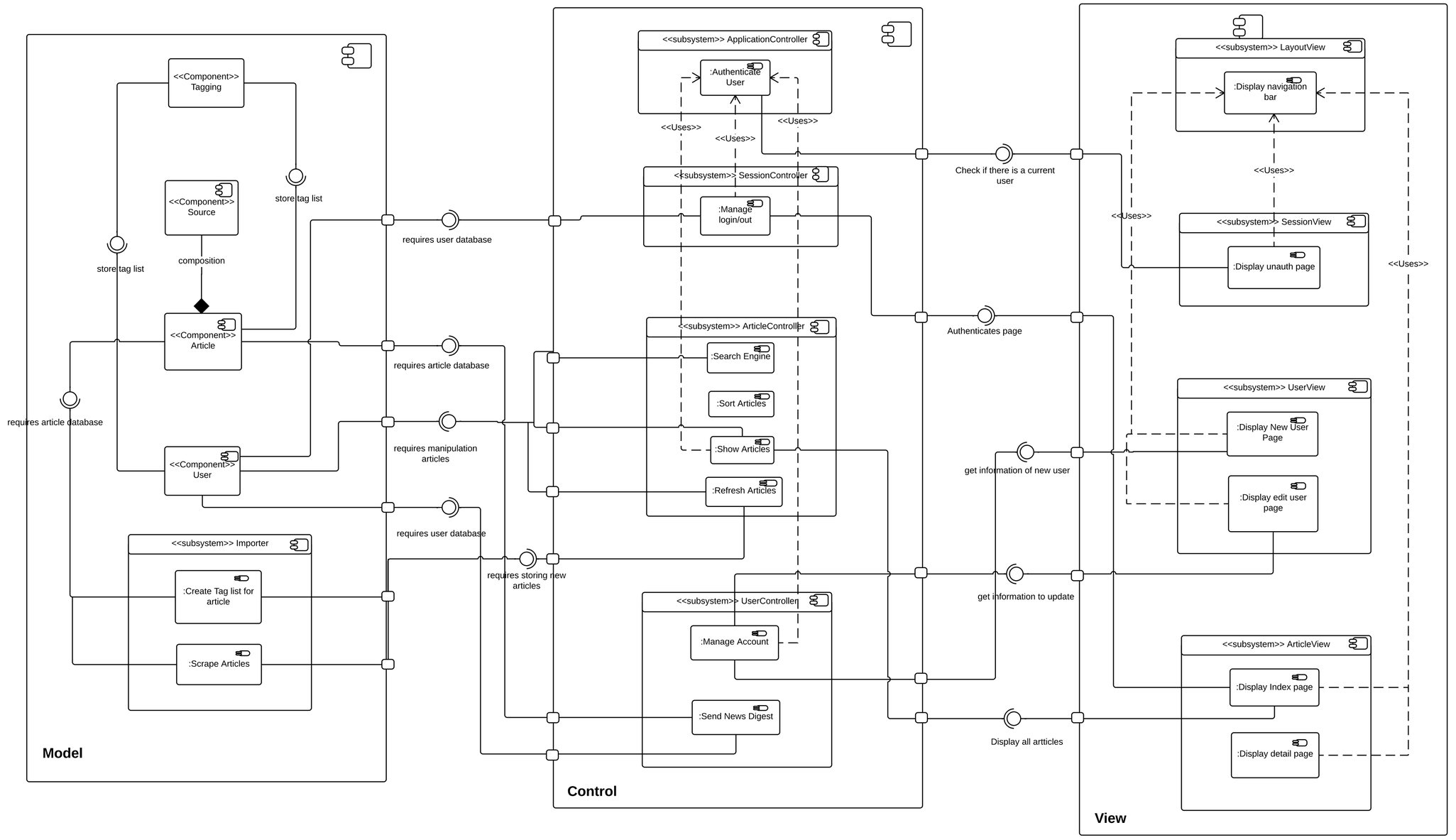
Group 13
Project 3
Sequence Diagram - tagging of articles
Group member:
Tessa Song: 597952
Nan Lin: 651025
Nihal Mirpuri: 562439



Group 13
Project 3
Sequence Diagram -
Compiling and emailing of a news digest
Group member:
Tessa Song: 597952
Nan Lin: 651025
Nihal Mirpuri: 562439



Group 13
Project 3
Component Diagram
Group member:
Tessa Song: 597952
Nan Lin: 651025
Nihal Mirpuri: 562439



A Short Design Rational for Class Diagram/Sequence Diagram/Components Diagram

*** Introduction to our design, assumptions and control structure**

For the user session, we allow user to create account with their personal information under the user class and store those information in the database. There are some restriction for particular information like password,email and user in order to prevent from duplication and privacy issue .We have a user controller class which allow users to create, edit, update and destroy their account. We also have a session controller class to authenticate whether the user is the current registered user. User can simply login with their username and password after registered.

For the article session, we have an importer class refers to the detailed information(like title, public date, author, image, summary, link, source, etc.) of articles which are getting from the source class. Then, we will display those information on the website page through an article controller class. We also have a scraper class to retrieve article update from different resources and displays those new updated articles into the webpage. There is a sorted function under the controller class, all articles will display in order and sort by the public date and time, and they will be stored only once in database. If they have the same article title, the duplicate one will be discarded. From the project specification, it says 'All of the news feeds must be scraped when the URL path /admin/scrape/ is requested. This URL must not require a login.' Therefore, we considerate to implement a scrape button with a path 'admin/scrape' and allow administrator to check update from the website link instead of using the scrape button in general. Then, we come out two design idea here. One of our idea is to restrict those functions for only registered user to see current articles and the new updated articles by press the scrape button. The other one is to show articles and update to all registered and non-registered users automatically by administrator. We feel a bit confused with this point but we think the second idea is more reasonable to implement. One more important thing that we want to specify here is the relationship between user and the scarper (uses for updated the newest articles). We discuss this problem in group whether an instance of ArticleController will run unique to each user (i.e. Search Engine, refresh() etc.). However we decided that returning articles to users should not be modified by individual user data, rather by only the interests or keywords in their search. Thus we do not need to show the association between Users and ArticleController.

For the article tagging, we have five method to tag an article based on its title summary/ description, keywords/category, author and source name which are shown as five class inherit from the main article tagging class in the diagram. We will use and display the results for combining the five method for each article. We will also remove the redundant words that are finding from different methods for the tag. We have considered to represent these five tagging method to five module instead of five class in the diagram. However, we have tried to find resource about how to add module into class digram and we did not find any useful information. Thus, we decided to use classes to represent them to avoid any confusion in the diagram. Based on the specification requirements, we also separate scraping module and the tagging module into two independent module to implement.

For the article searching, we will search articles by the keyword and display the first 10 articles based on the score of their weight in descending order and implement this function under the article controller class. The weight calculation will consist of the following situations. If the keyword matches the article tags/title/description/source, it will get 4/3/2/1 points in the total weight. If the articles have the same weight in matching search, they will be sorted by the published date and time and the most recent articles will be displayed.

For the compiling and emailing a new digest session, the article will be sent to subscribers' email based on their request and interest. Users can update their interested article area as a keywords which is same as a article tag in the profile page. So we have a tagging class which match users' interests with their interested articles. The design of the interests will be like if the user interested in music and put music as one of his/her interests, we may return articles tagged as "music", "Music", "classical music", "pop music", etc.. Currently, what we have done is only the exactly match articles will be shown to users and we need to do the further improvement. We also consider to add an checkbox under the user profile page and provide user to choose whether they want us to send them those articles that they feel interested in to their registered email. The implementation will be under the user controller class. We have two function called 'send_new_digest' and 'interesting_articles'. When we found users' interested articles by matching the article tag, we will use an array to store the first ten matching article in the 'interesting_articles' function based on their publish date and time (most recent first) and then send to users' registered email by using the 'send_new_digest' function.

The design we chose implements a decentralised/distributed control structure, which is evident through the use of ArticleController to manage the articles, UserController to manage the users, etc. (i.e. There is no central controller managing the server). This allows us to separate the modules to allow for greater flexibility.