BSc Project

Computer science Department

Project ID: UQU-CS-2019F-29

December 2019

Mining Information and Discover knowledge from Microblogs Data

Imprint

Authors:

	Name	ID	Email
1	أثير ماجد الحازمي	436000389	s436000389@st.uqu.edu.sa
2	إلهام مسفر الاسمري	436001115	s436001115@st.uqu.edu.sa
3	شهد عماد السيد	436039450	s436039450@st.uqu.edu.sa
4	ليان صالح الاحمدي	436001850	s436001850@st.uqu.edu.sa
5	هديل يوسف الردادي	436000308	s436000308@st.uqu.edu.sa

Supervisor: Dr. Loui Alarabi.

Dept. of Computer Science.

Faculty of Computer and Information Systems.

Umm Al-Qura University, KSA.

Intellectual Property Right Declaration

This is to declare that the work under the supervision of " Dr. Loui Alarabi "having title “ imprint system” carried out in partial fulfillment of the requirements of Bachelor of Science in Dept. of Computer Science , is the sole property of the Umm Al Qura University and the respective supervisor and is protected under the intellectual property right laws and conventions. It can only be considered/ used for purposes like extension for further enhancement, product development, adoption for commercial/organizational usage, etc., with the permission of the University and respective supervisor. This above statement applies to all students and faculty members.

Date: 20/4/2020

Author(s):

Name: أثير ماجد الحازمي

Name: إلهام مسفر الاسمري

Name: شهد عماد السيد

Name: ليان صالح الاحمدي

Name: هديل يوسف الردادي

Supervisor(s): Name: Dr. Loui Alarabi

ACKNOWLEDGEMENT:

In the first place we thank Allah for everything he blessed us with. We would like to express our special thanks to Dr. Loui Alarabi our research supervisor who give his knowledge, advice in kindly and professional way to take our efforts to the next step, his patience, and his unforgettable encouragement. we thank our families and our siblings for their valuable care, patience, kindness, and supporting us through this work.

ABSTRACT:

this document explains proposed imprint system, well web designed system for querying, analyzing tweets microblogs. Treat tweet microblogs which consider as stream of rich data that carries different types of information. it has three main operations: locate hashtag, show trend evolution, Cluster account by their interest. locate hashtag specify location of origin tweet, show trend evolution that describe how trend spread along rang of distance and specify these locations in addition display other information like duration upon these locations, Cluster account by their interest by using text classification determine which the most topics user write about them. And presents the result on a map interface that allows an interactive vision.

Keywords: Twitter, cluster, text classification, tweet.

TABLE OF CONTENTS: v

Chapter 1 INTRODUCTION

1.1 Purpose of This Document

the purpose of this documentation is to describe the Imprint system and its operations which assist in mining information and discover knowledge from microblogs datasets to provide a description of how information start, show its evolution, specify its origin and determine its location for meet information seekers needs.

1.1.1 Brief Description

The proposed system idea come from the 2030 vision which needs to level up many respects such as: Volunteering, E-Government, and cybersecurity. For instance, when a trending topic takes a place in time to distribute fake news, authority agencies might be interested to cluster these accounts and mine deeper information about their user-based, influence or even their location. The current efforts are either does not offer a deeper knowledge such as: how information starts? How it spread and its origin?[1], or they are mainly focused on enhancing the algorithms to solve a specific problem [6,7,8]. Unlike our system, which is a novel that provides an extendable framework that can house some data mining and analysis algorithms and display results in an understandable way.

1.2 Purpose of The Project

1.2.1 Project Scope

The scope of the project is in the area of database, and data mining. In this project we will build a system that is equipped with a set of text classification techniques. The system will be very useful for stakeholders, private organizations, and government. As they will assist them in mining information and discover knowledge from microblogs datasets. Specifically, from a Twitter data stream.

1.2.2 Outcome

The system with a nicely designed web-interface analyzes the twitter dataset. The user will interact with the web-interface. The web-interface will allow users to submit the following requests:

1. Locate analyzed twitter hashtag.
2. Give a list of accounts that fit the search query.

3. Apply Data mining algorithms to discover information.
4. Cluster users based on their interests.
5. Shows the evolution of trending topics.

Chapter 2 BACKGROUND

2.1 Background

Social media services have become very popular in the last decade which has led to explosive growth in size of microblogs data, which consider as stream of rich data that carries different types of information including text, location, photos, and language information[1,10] which led to rise up the need of analyzing these microblogs data, this information and its analysis can establish a margin and organize how people are thinking or reacting toward a specific topic or how they are interested in a particular domain. To analyzing and mining data from microblogs and reused it in its new shape to achieve specific goals it is hard work, but the result is worth it. in this research our focus on analyzing Twitter microblogs data that allows people to communicate with short, 280-character messages that roughly correspond to thoughts or ideas known as tweet which consider as basic atomic building block of all things on Twitter[8,10], contains a lot of information as picture, video, link, location, and text. tweets fall in kinds or categories or topics “hashtag”, it is words prepended by “#” (hash symbols), main features of Twitter shows a list of top hashtags so-called trending topics at all times, they reflect the topics that are being discussed most in the latest minutes on the site’s stream of tweets either in the world, on TV, or on the Internet, can motivate users to discuss on a topic. Trending topics have generated big interest not only for the users themselves but also for information seekers [3]. in addition of explicit infrastructure of Twitter, its provides some features to researchers, it is considered as a platform for brainstorming ,gather Twitter data, as it is a reach and it comes with a low-cost, rich developer tooling, and broad appeal to users from every walk of life[10].

2.2 Existing System

2.2.1 Existing System Description

[Tagreed]:

Taghreed is characterized by a large number of users of the system queries and the results are presented on a map and provides a user interface. The amount of interaction of microblogging and the memory contents are restored from backups. Taghreed's design principles are based on:

[1] Dominance of the temporal, spatial, and keyword which is promotes these attributes as richest attributes in microblogs which provides effective indexing for search queries.

[2] Importance of queries on recent microblogs. it is supporting the queries with consideration of the real time and all subsequent requirements of main-memory indexing, flushing management, and recovery management.

The main technical challenges in Taghreed comes from three main sources: (a) the large number of microblogs, (b) the continuous arrival of microblogs with high rates (thousands per second), and (c) the new types of queries on the rich microblogs data, Taghreed main components are: (1) Efficient and scalable data indexing, (2) efficient interactive query processing, (3) low-overhead recovery management, and (4) effective data visualization [1].

[TwitterStand]:

Twitter stand it is concerned with news extraction and organize them as a traditional setup of reporters and newspapers. it is extract news from twitter microblogs by the user's tweets to be the main block of building the system either they provide their opinions or from the original new. The challenge is to identify, extract, filter news tweet in real time from non-news tweets while managing with a massive amount comes of input data and displays them according to their geographic location, and the importance of the news topic. providing a map interface for reading this news [8].

[product sentiment analysis]:

This search deal with twitter sentiment analysis which become the most interesting and important topic in marketing research due to its promising commercial benefits to investigate public opinion regarding some product in order predict products overtime. its goal to predict people's thoughts about some devices according to set of tweets collected to make compare device features like camera, battery. Approach tracked two steps: first, preprocessing twitter data in order to filter noisy data which show how they can improve the system accuracy but it is often underestimated and not extensively covered in the literature, Second, Classification technique to label data to positive or negative in order to predict people's opinion[6].

[online Arabic content analysis dashboard]:

this system stands for acquire Twitter data according to user's search keywords to study public opinion on many topics, analysis sentiment Arabic tweets, display the classify of the keyword, and compare the current results with results in other years. its goal to analyzing data to understand them more clearly and help the decision-maker to reach better results to work efficiently, increase the predictability of planners, and improve the processes of production and operation in the fields of service and productivity [7].

2.2.2 Problems in The Existing System

all these previous studies concern about how to analyze the data or categorize people opinion of some product to negative or positive, or analyzing data microblogs of Arabic tweets,

Tagreed unable to mine or discover information from microblogs data. Similarly, TwitterStand is capable of extracting news and making report; yet it is ill to acquire information and discover knowledge from twitter data [1].

2.2.3 Summary:

In this chapter define twitter entities, define the scope of this project which related to data mining the result will be displayed in a well-designed web interface. List existing systems and their features, challenges, and in last explain their problems.

Chapter 3 ANALYSIS AND DESIGN

3.1 Data Analysis

3.1.1 Functional Requirements

3.1.1.1 Users' Requirements

1. User shall be able to cluster accounts by given interest.

2. User shall be able to show the origin location of specific hashtag.

3. User shall be able to show the evolution of trending topics.

3.1.1.2 System Requirements

1. Cluster account based on their interests

1.1. System shall be able to cluster accounts by given interest.

1.2. System shall be able to list the account.

2. Locate twitter hashtag

2.1. System shall be able to analyze spatial data.

2.2. System shall be able to specify the location.

3. Shows the evolution of trending topics.

3.1. System shall be able to track trend evolution.

3.2. System shall be able to analyze spatial.

3.3. Systems shall be able to specify locations.

3.4. System shall be able to display results on histogram.

3.1.2 Non-Functional Requirements

1. **Usability:** system shall be easy to learn and use, users have become familiar with interface design, users shall be able to remember how to use it once they learned, the website makes it easy for the experienced user to quickly complete tasks.
2. **Performance:** A website should be capable enough to handle 10 thousand requests without affecting its performance, response time should be reasonable depending on the tasks and how much time it takes.
3. **Portability:** means the ability to run the software on any platform either Windows, Mac, or Linux. Python is software written in C language and take portability feature from it. Moreover, Python offers multiple options for coding portable graphical user interfaces, database access programs, web-based systems, and more.

3.1.3 Proposed Solutions

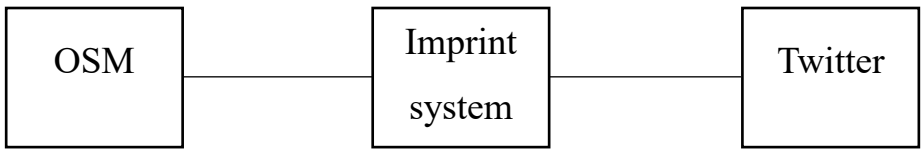
we propose imprint system to track data about when, and where information generated or released on twitter. it is useful to detect how people are interested in special scope and show how information can be spread along range of distance and specify these locations. it facilitates the search process and obtain these queries in a well-designed website.

3.1.4 Alternative Solutions

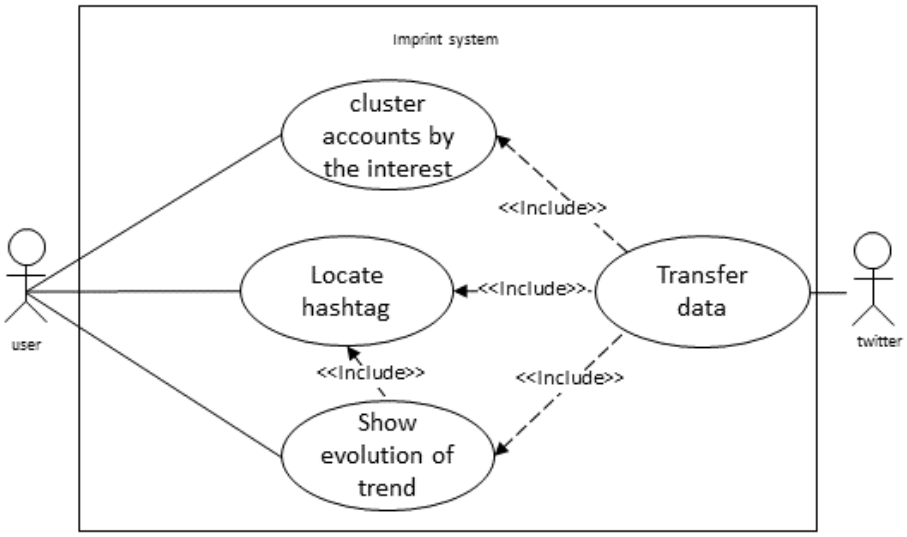
As mentioned in chapter two existing systems addresses only specific services, they are unable to discover or having limited track of data about how, when, and where information generated on twitter. Twitter allow their users to search for specific hashtag and account, but Twitter can't determine where and when the origin of hashtag generated or show evolution of trending topics or search for an account in a specific scope. This website will fill this gap.

3.2 UML System Models

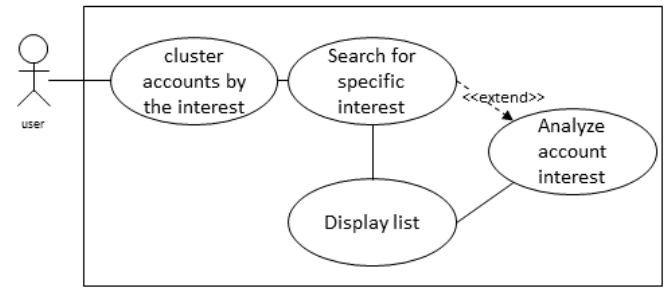
3.2.1 Context Models

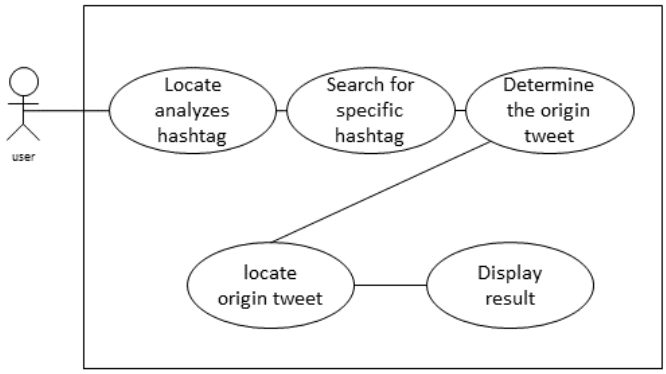


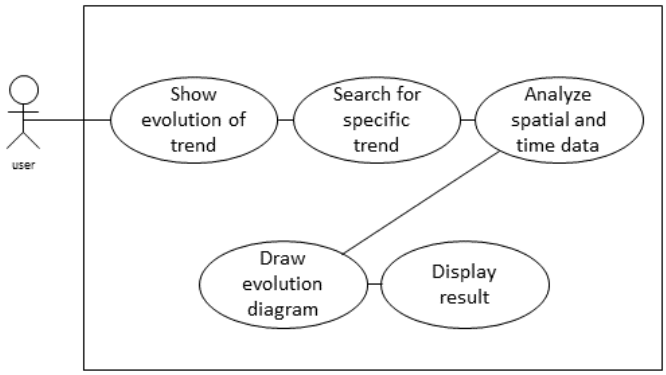
3.2.2 Use Case



3.2.2.1 Sub Use Case







3.2.3 Use Case Scenario

Use Case Name Transfer data.	ID UC-04.	Priority high.
Actor: Twitter.		
Description: Transfer unstructured information from twitter API to the imprint system to use it, and storage it in a file.		
Trigger: this use case is initiated when order the information from twitter. Type: external.		
Preconditional:		
Steps: 1- Transfer data from twitter. 2-conver data from Jason to CSV.		
Alternative course:		
Post condition: - Twitter data stored in temporal file.		
Summary:		

Inputs	Source	output	destination
Data	System	twitter data.	File system.

Use Case Name cluster accounts by the interest.	ID UC-A.	Priority high.
Actor: user.		
Description: user can search for specific interest the system shall be cluster all accounts that fit the search query and display the list to the user.		
Trigger: this use case is initiated when a user interest is submitted, Type: external.		
Preconditional: --		
Steps: 1- user chose specific interest . 2-submit.		
Alternative course:		
Post condition: - Accounts that interest in the same domain.		
Summary:		

Inputs	source	output	Destination
Interest.	User.	Clustered accounts depend on the user inputs.	Database. User.

Use Case Name Analyze account interest	ID UC-A2.	Priority high.
Actor: user.		
Description: Extract all tweet in the account then Analyze based on interest and save result		
Trigger: this use case is initiated when a user Click interest button. Type: external.		
Preconditional: --		
Steps: 1- after cluster do analyze account interest.		
Alternative course:		
Post condition: - Save result.		
Summary:		

Inputs	source	Output	Destination
All tweet	User.	Interest for all account.	Database . User.

Use Case Name Search specific interest	ID UC-A1.	Priority high.
Actor: user.		
Description: user can search for specific interest the system shall be cluster all accounts that fit the search query		
Trigger: this use case is initiated when a user Click interest button. Type: external.		
Preconditional: --		
Steps: 1- a user Click interest button 2-the system do cluster process		
Alternative course:		
Post condition: - Show list of account		
Summary:		

Inputs	source	output	destination
Specific interest	User.	Show list account based on interest	Database . User.

Use Case Name Display list	ID UC-A3.	Priority high.
Actor: user.		
Description: user can search for specific interest the system shall be cluster all accounts that fit the search query and display the list to the user.		
Trigger: this use case is initiated when a user Click interest button. Type: external.		
Preconditional: --		
Steps: 1- display all account interest .		
Alternative course:		
Post condition: - Show list account.		
Summary:		

Inputs	source	output	Destination
Choose interset	User.	List Account	Database . User.

Use Case Name: Locate hashtag.	ID UC-B.	Priority high.
Actor: user.		
Description: user can ask the system for specific hashtag, it extract origin which means the oldest tweet in the hashtag and specify its location, the result display in map interface.		
Trigger: this use case is initiated when a user submit specific hashtag. Type: external.		
Preconditional: --		
Steps: 1- user chose hashtag. 2-submit.		
Alternative course:		
Post condition: - origin of the hashtag, spread of this hashtag.		
Summary:		

Inputs	source	output	destination
hashtag.	User.	origin of the hashtag, spread of this hashtag.	Database. User.

Use Case Name Locate analyzes hashtag	ID UC-B1.	Priority high.
Actor: user.		
Description: it extract origin which means the oldest tweet in the hashtag and specify its location		
Trigger: this use case is initiated analyzes locate hashtag Type: external.		
Preconditional: --		
Steps: 1- System do analyzes hashtag. 2-then locate hashtag.		
Alternative course:		
Post condition: - Show origin in the hashtag.		
Summary:		

Inputs	source	output	Destination
Specific hashtag.	User.	Origin hashtag	Database . User.

Use Case Name Search for specific hashtag	ID UC-B2.	Priority high.
Actor: user.		
Description:user can ask the system for specific hashtag then Determine the origin tweet		
Trigger: this use case is initiated when a user choose specific hashatg Type: external.		
Preconditional: --		
Steps: 1-Search specific hashtag in data. 2-the system do Analyze.		
Alternative course:		
Post condition: - Show origin in the specific hashtag.		
Summary:		

Inputs	source	output	destination
Choose hashtag	User.	Origin hashtag	Database . User.

Use Case Name Determine the origin tweet	ID UC-B3.		Priority high.
Actor: user.			
Description:user can ask the system for extract origin of tweet which means the oldest tweet and specify its location			
Trigger: this use case is initiated when choose Specific hashtag. Type: external.			
Preconditional: --			
Steps: 1- after analyze Determine the origin tweet.			
Alternative course:			
Post condition: - The origin			
Summary:			
Inputs	source	output	destination
Specific hashtag.	User.	origin tweet	Database . User.

Use Case Name locate origin tweet	ID UC-B4.		Priority high.
Actor: user.			
Description: it extract origin which means the oldest tweet in the hashtag and specify its location			
Trigger: this use case is initiated when choose Specific hashtag. Type: external.			
Preconditional: --			
Steps: 1- extract first tweet in the hashtag. 2-Determine location.			
Alternative course:			
Post condition: - Show specific hashtag in map			
Summary:			
Inputs	source	output	destination
Specific hashtag.	User.	Show location hashtag on map	Database . User.

Use Case Name Display result	ID UC-B5.	Priority high.
Actor: user.		
Description: user can ask the system for specific hashtag, it extract origin which means the oldest tweet in the hashtag and specify its location, the result display in map interface.		
Trigger: this use case is initiated when a user Click Search button. Type: external.		
Preconditional: --		
Steps: 1- Display specific location in map.		
Alternative course:		
Post condition: - Show specific hashtag in map		
Summary:		

Inputs	source	output	destination
Specific hashtag.	User.	Show location hashtag on map	Database . User.

Use Case Name Show evolution of trend	ID UC-C.	Priority high.
Actor: user.		
Description : system provide description of specific trending topic.		
Trigger : this use case is initiated when click trending topic. Type: external.		
Preconditional: --		
Steps: 1- user chose trending topic. 2-submit.		
Alternative course:		
Post condition: Show evolution trend		
Summary:		

Inputs	source	output	destination
trending topic.	User.	Show evolution trend	Database . User.

Use Case Name Search for specific trend	ID UC-C1.	Priority high.
Actor: user.		
Description:- the system Search for specific trend.		
Trigger: this use case is initiated when click trending topic. Type: external.		
Preconditional: --		
Steps: 1-Search specific trend in data.		
Alternative course:		
Post condition: Show evolution trend .		
Summary:		

Inputs	source	output	destination
trending topic.	User.	Show evolution trend .	Database . User.

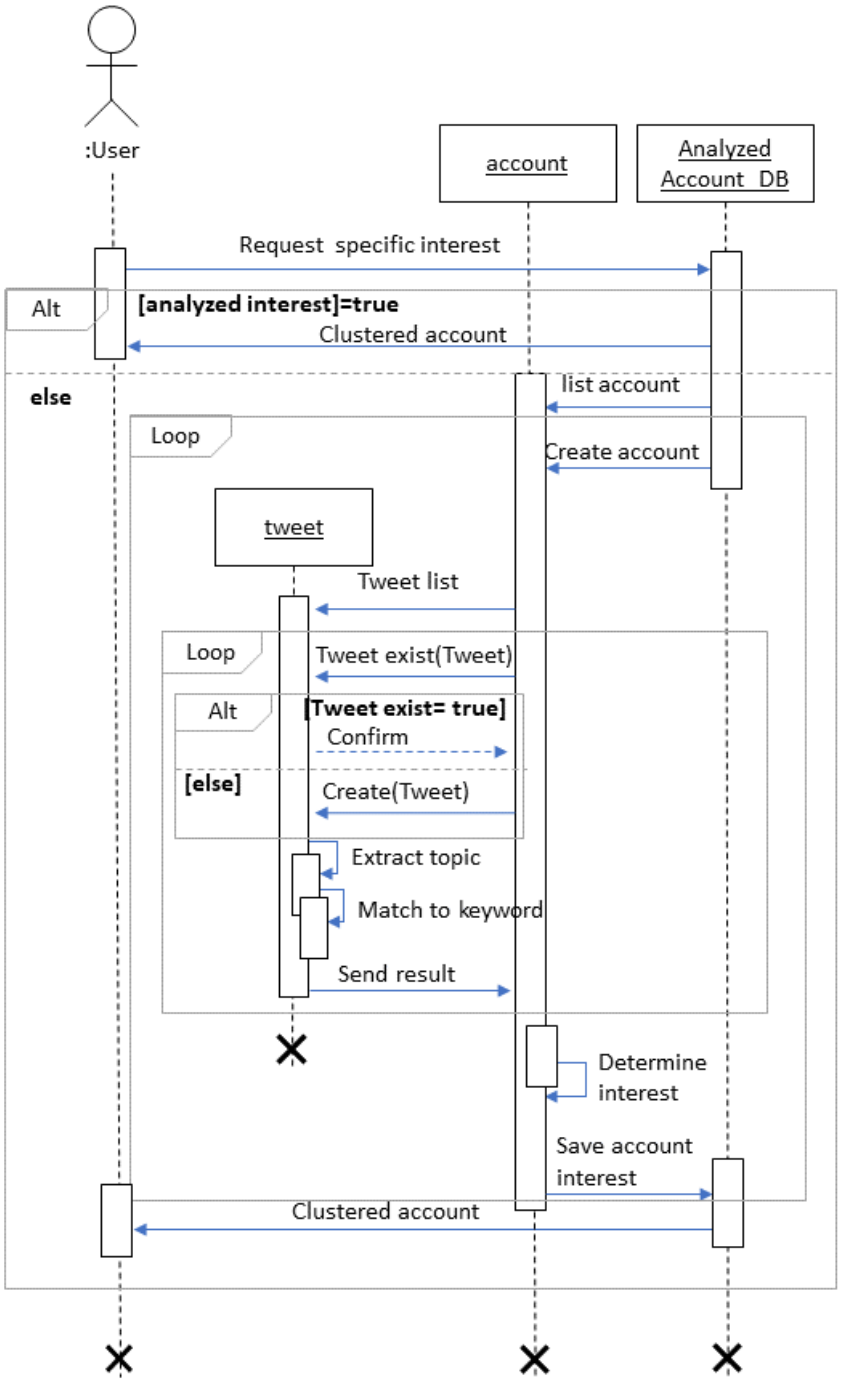
Use Case Name Draw evolution diagram	ID UC-C3.	Priority high.
Actor: user.		
Description: the System Draw evolution diagram basic in trend topic .		
Trigger: this use case is initiated when click trending topic. Type: external.		
Preconditional: --		
Steps: 1- Draw evolution diagram basic in trend topic .		
Alternative course:		
Post condition: -Draw evolution diagram		
Summary:		

Inputs	source	output	Destination
Data trending topic .	User.	Draw evolution diagram.	Database . User.

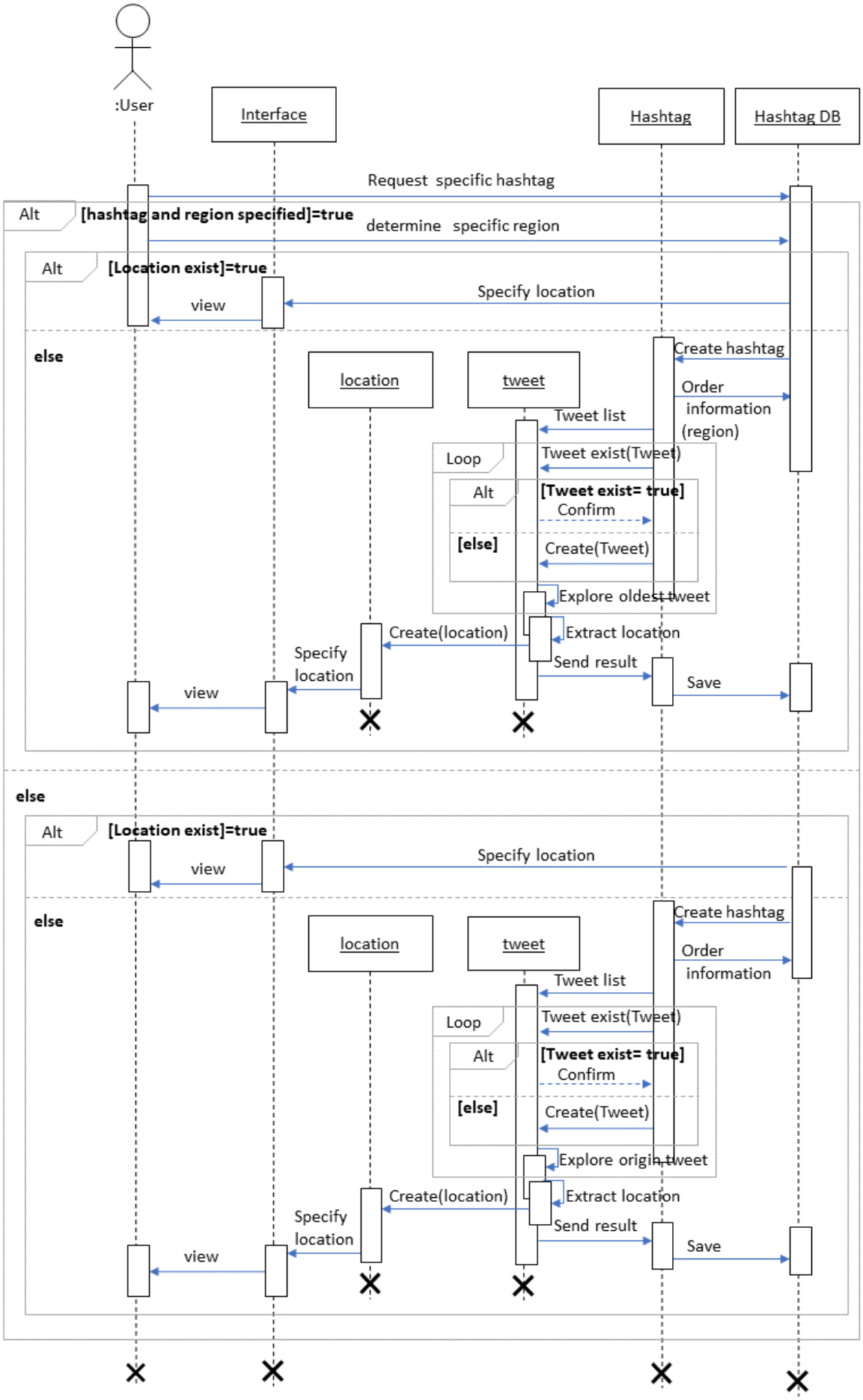
Use Case Name Display result	ID UC-C4.	Priority high.
Actor: user.		
Description : the system Display result for user.		
Trigger: this use case is initiated when click trending topic. Type: external.		
Preconditional: --		
Steps: 1- Display trending topic in diagram .		
Alternative course:		
Post condition: Display result .		
Summary:		

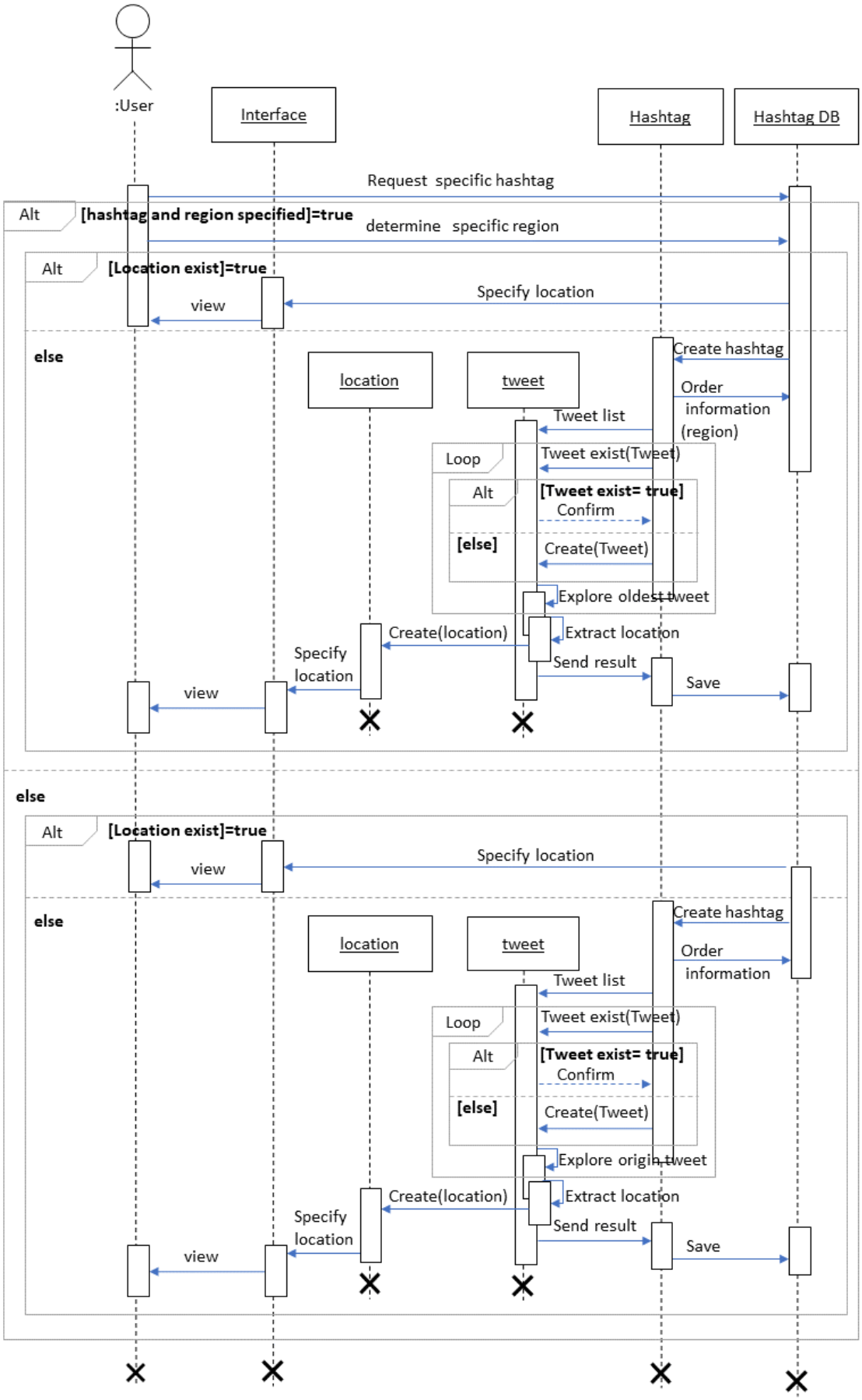
Inputs	source	output	destination
evolution diagram.	User.	Display result .	Database . User.

3.2.4 Sequence Diagram

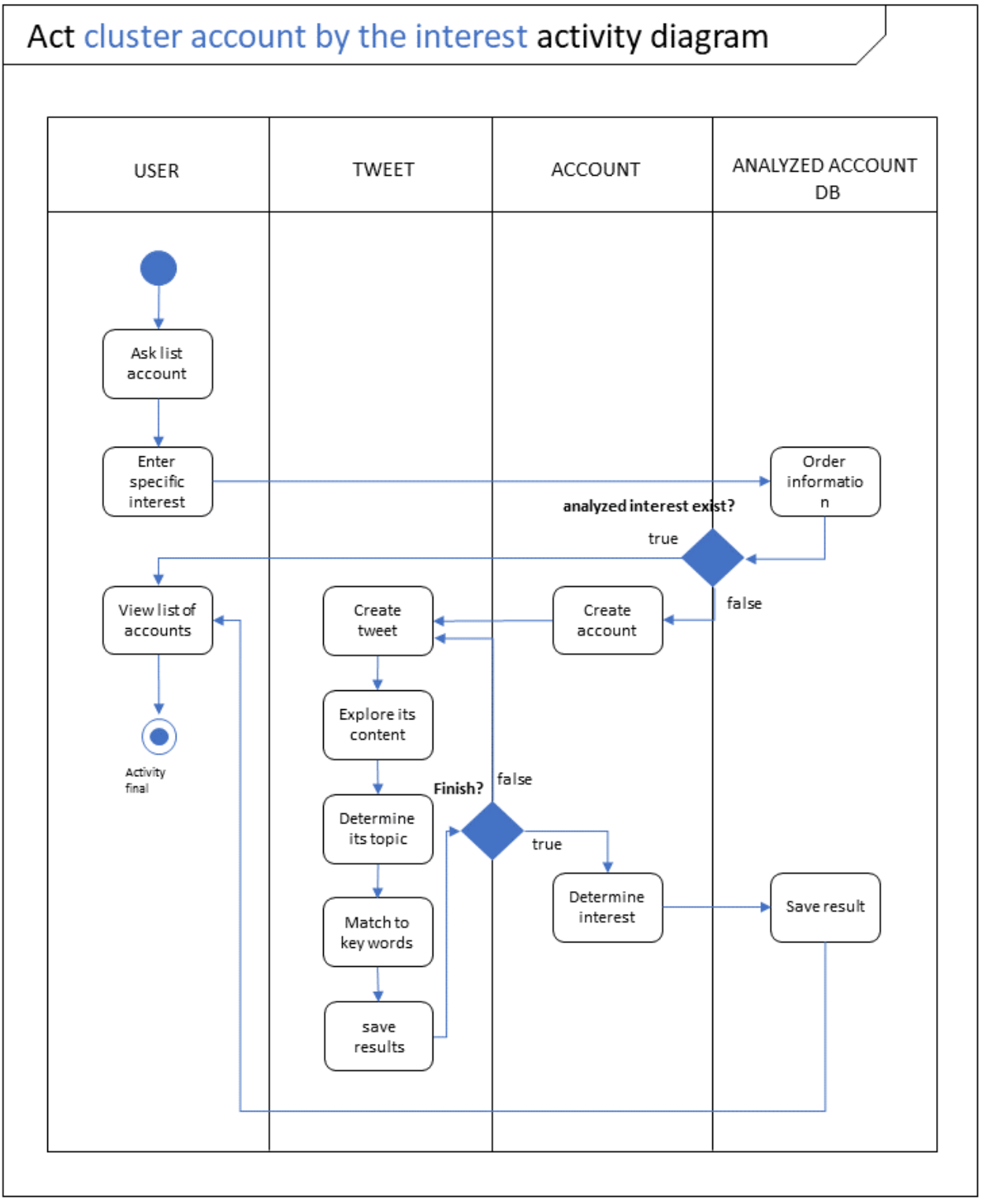


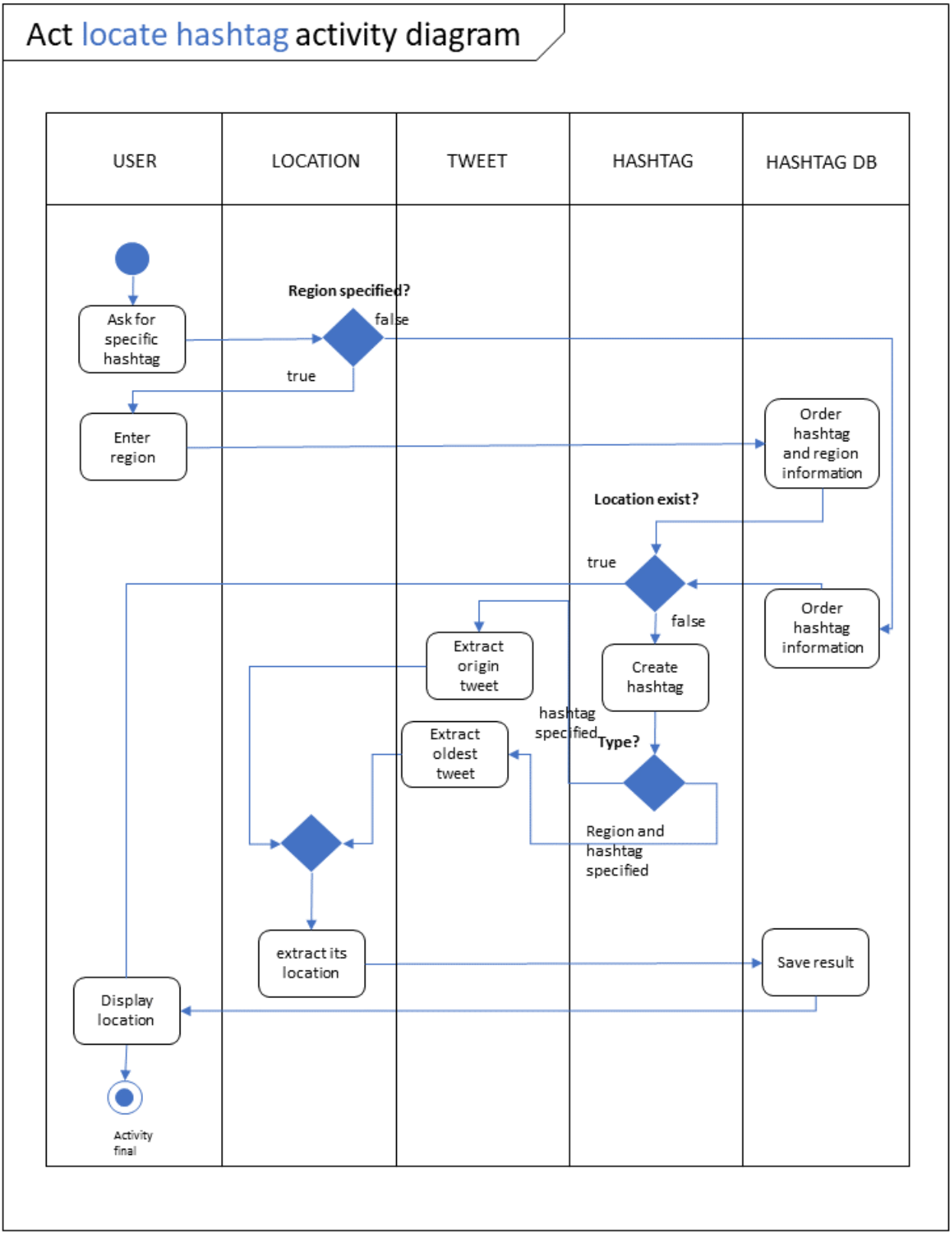
cluster accounts interest


:User
Interface
Hashtag
Hashtag DB
Request specific hashtag
Alt
[hashtag and region specified]=true
determine specific region
Alt
[Location exist]=true
Specify location
view
else
location
tweet
Create hashtag
Order information (region)
Tweet list
Loop
Tweet exist(Tweet)
Alt
[Tweet exist= true]
Confirm
[else]
Create(Tweet)
Explore oldest tweet
Create(location)
Extract location
Specify location
Send result
Save
view
else
Alt
[Location exist]=true
Specify location
view
else
location
tweet
Create hashtag
Order information
Tweet list
Loop
Tweet exist(Tweet)
Alt
[Tweet exist= true]
Confirm
[else]
Create(Tweet)
Explore origin tweet
Create(location)
Extract location
Specify location
Send result
Save
view

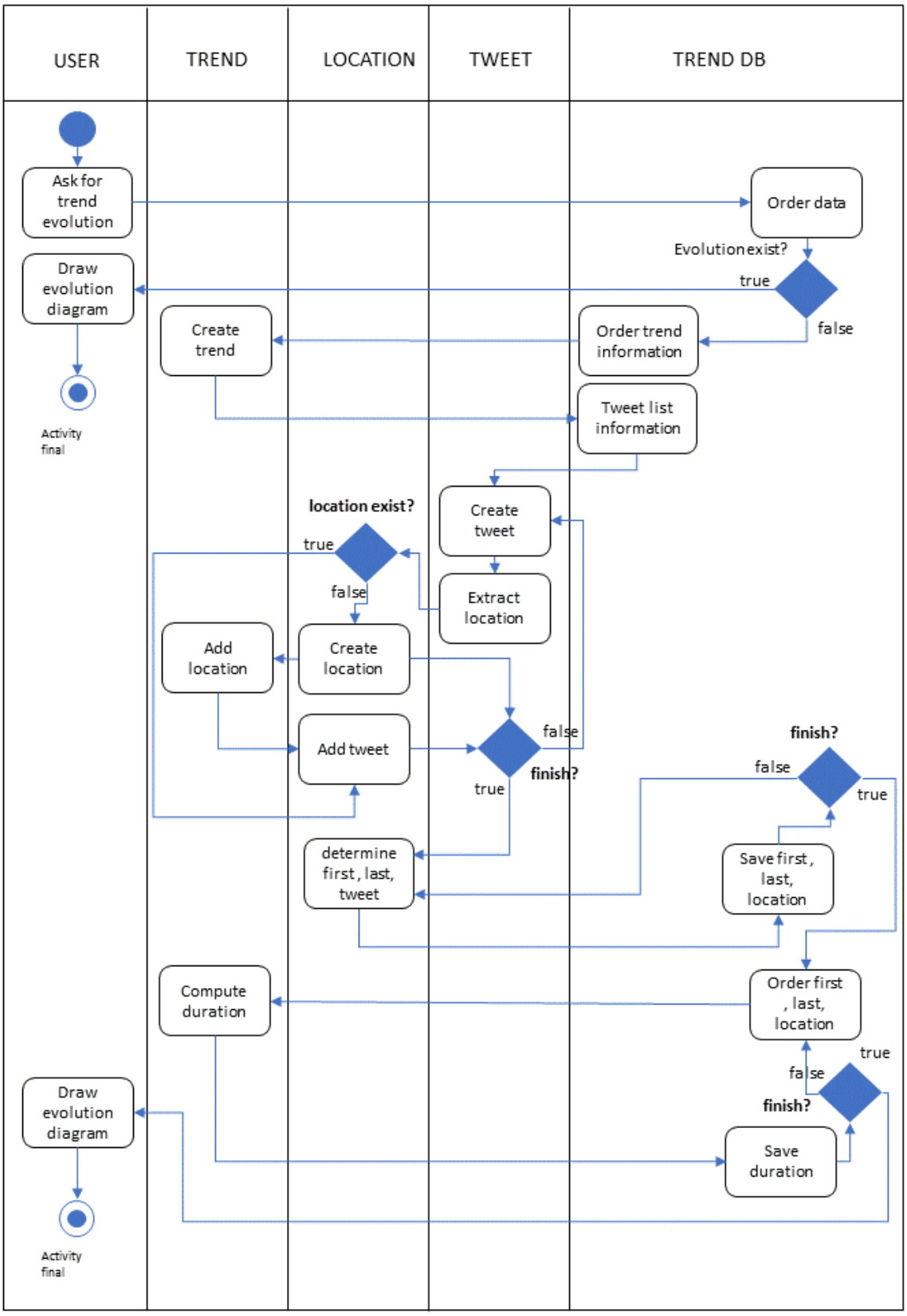

:User
Interface
Hashtag
Hashtag DB
Request specific hashtag
Alt
[hashtag and region specified]=true
determine specific region
Alt
[Location exist]=true
Specify location
view
else
Create hashtag
location
tweet
Order information (region)
Tweet list
Loop
Tweet exist(Tweet)
Alt
[Tweet exist= true]
Confirm
[else]
Create(Tweet)
Explore oldest tweet
Create(location)
Extract location
Specify location
Send result
Save
view
else
Alt
[Location exist]=true
Specify location
view
else
Create hashtag
location
tweet
Order information
Tweet list
Loop
Tweet exist(Tweet)
Alt
[Tweet exist= true]
Confirm
[else]
Create(Tweet)
Explore origin tweet
Create(location)
Extract location
Specify location
Send result
Save
view

3.2.5 Activity Diagram

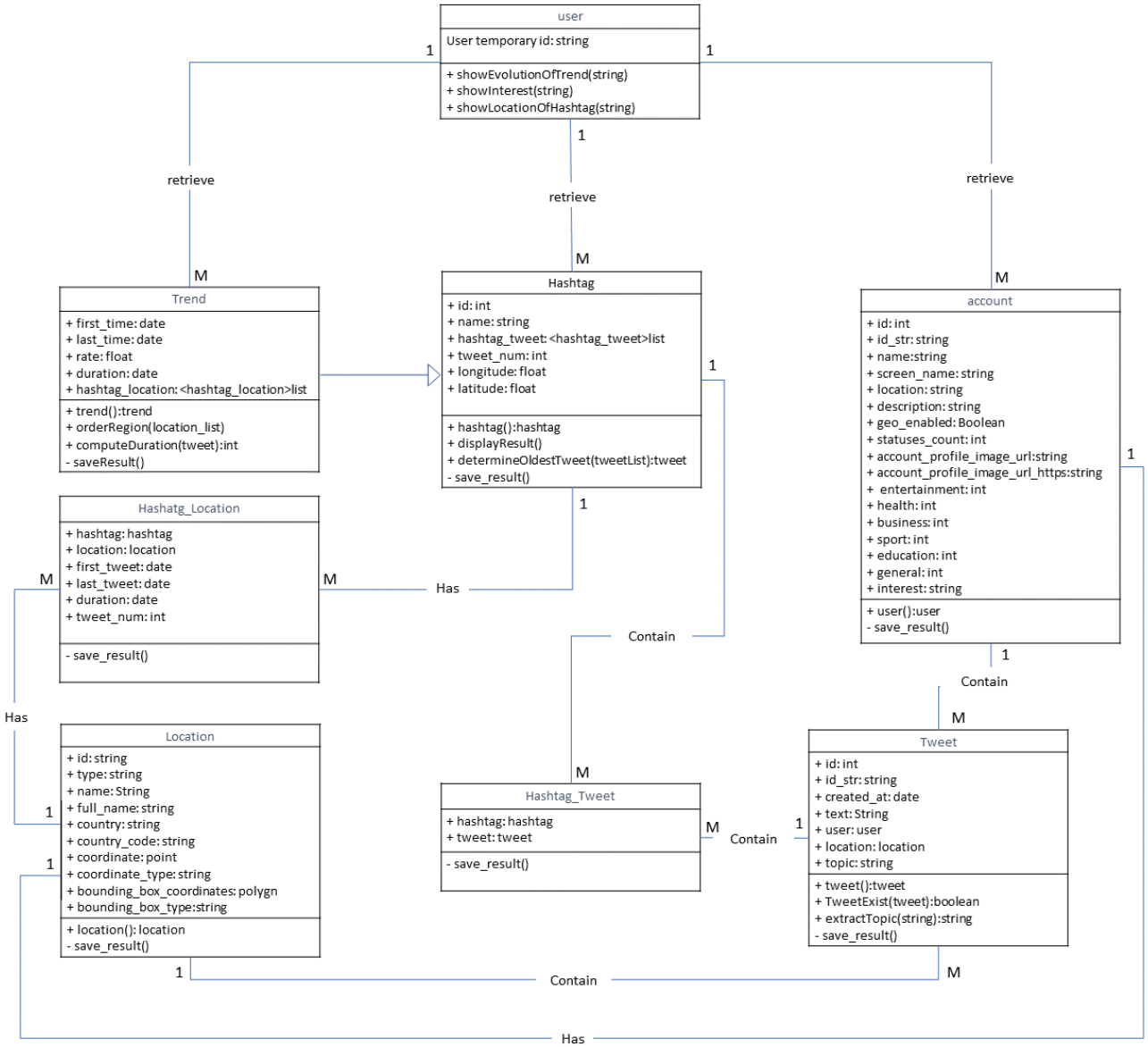



Act locate hashtag activity diagram
USER
LOCATION
TWEET
HASHTAG
HASHTAG DB
Ask for specific hashtag
Region specified?
false
true
Enter region
Order hashtag and region information
Location exist?
true
false
Order hashtag information
Extract origin tweet
Create hashtag
hashtag specified
Type?
Extract oldest tweet
Region and hashtag specified
extract its location
Save result
Display location
Activity final

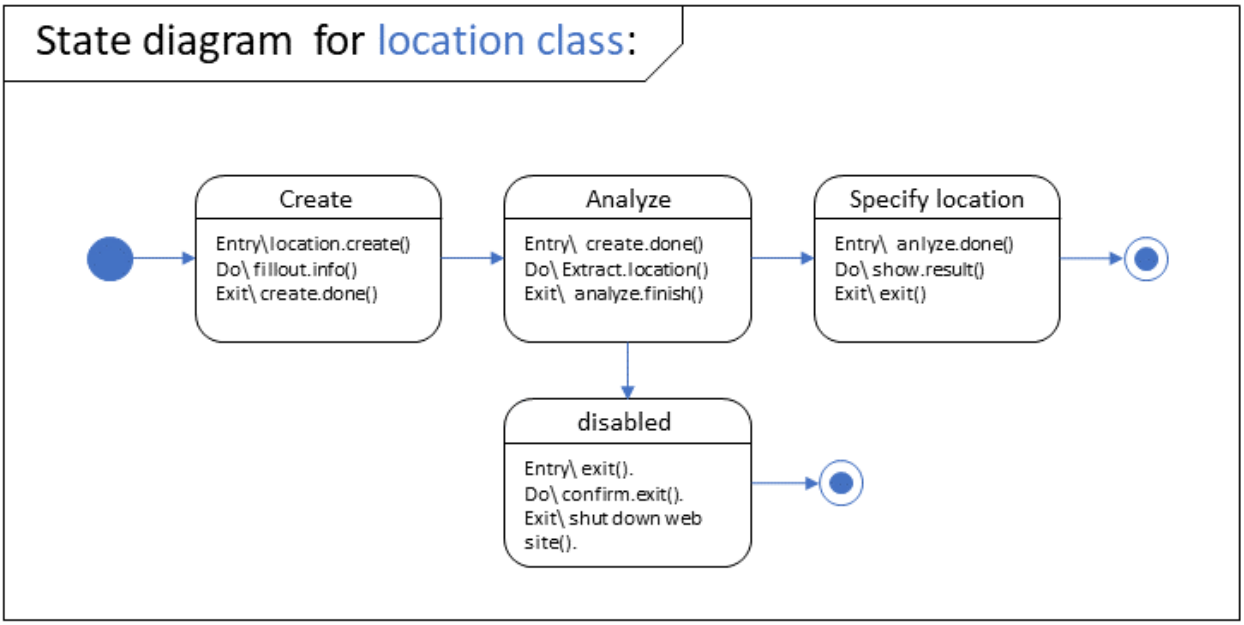
Act show trend evolution activity diagram

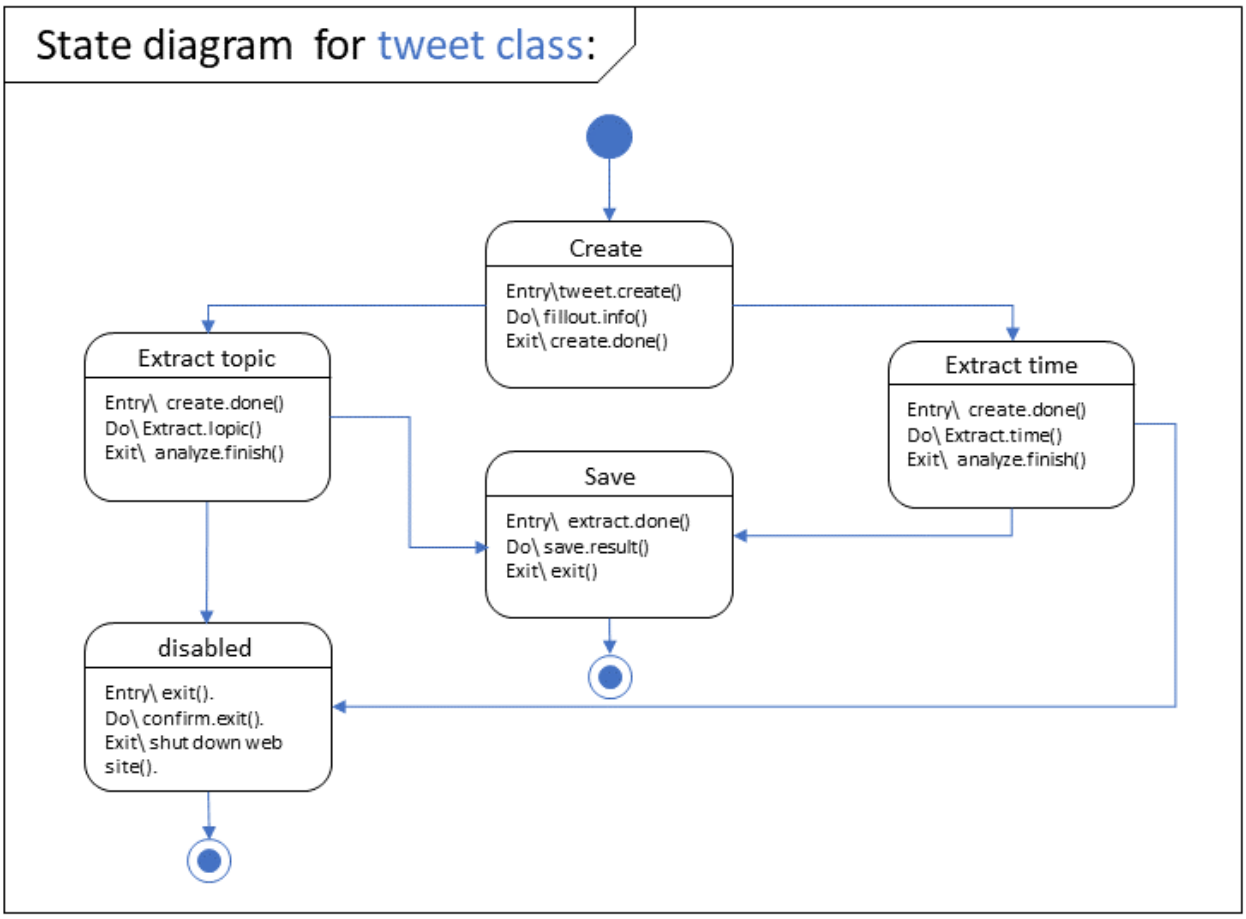


3.2.6 Class Diagram

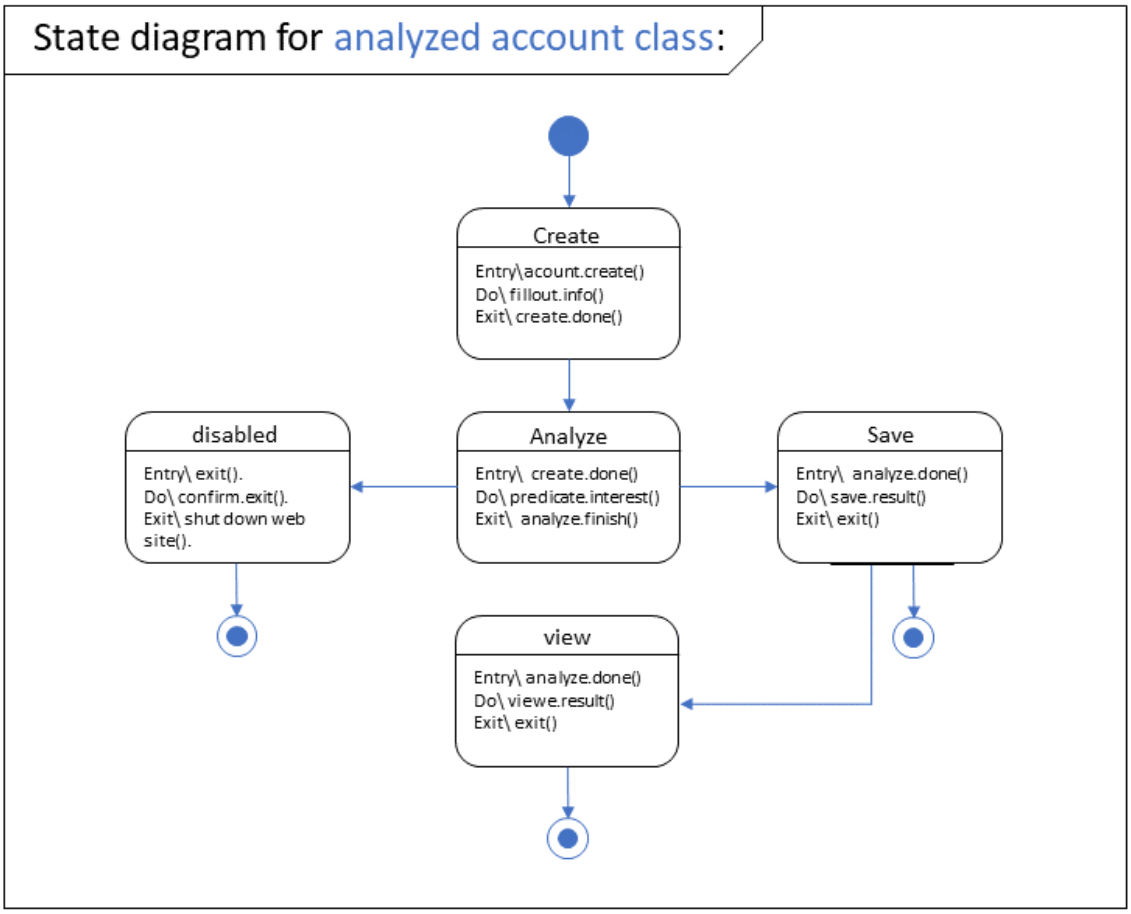


3.2.7 State Diagram

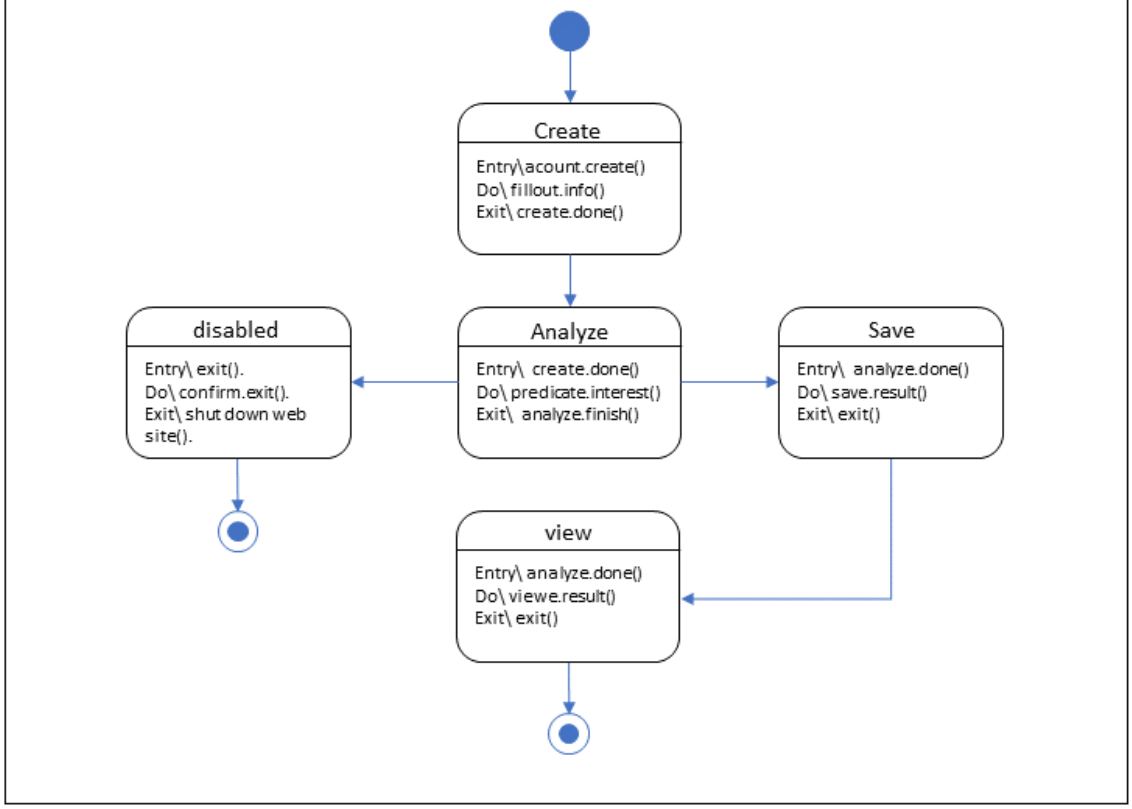




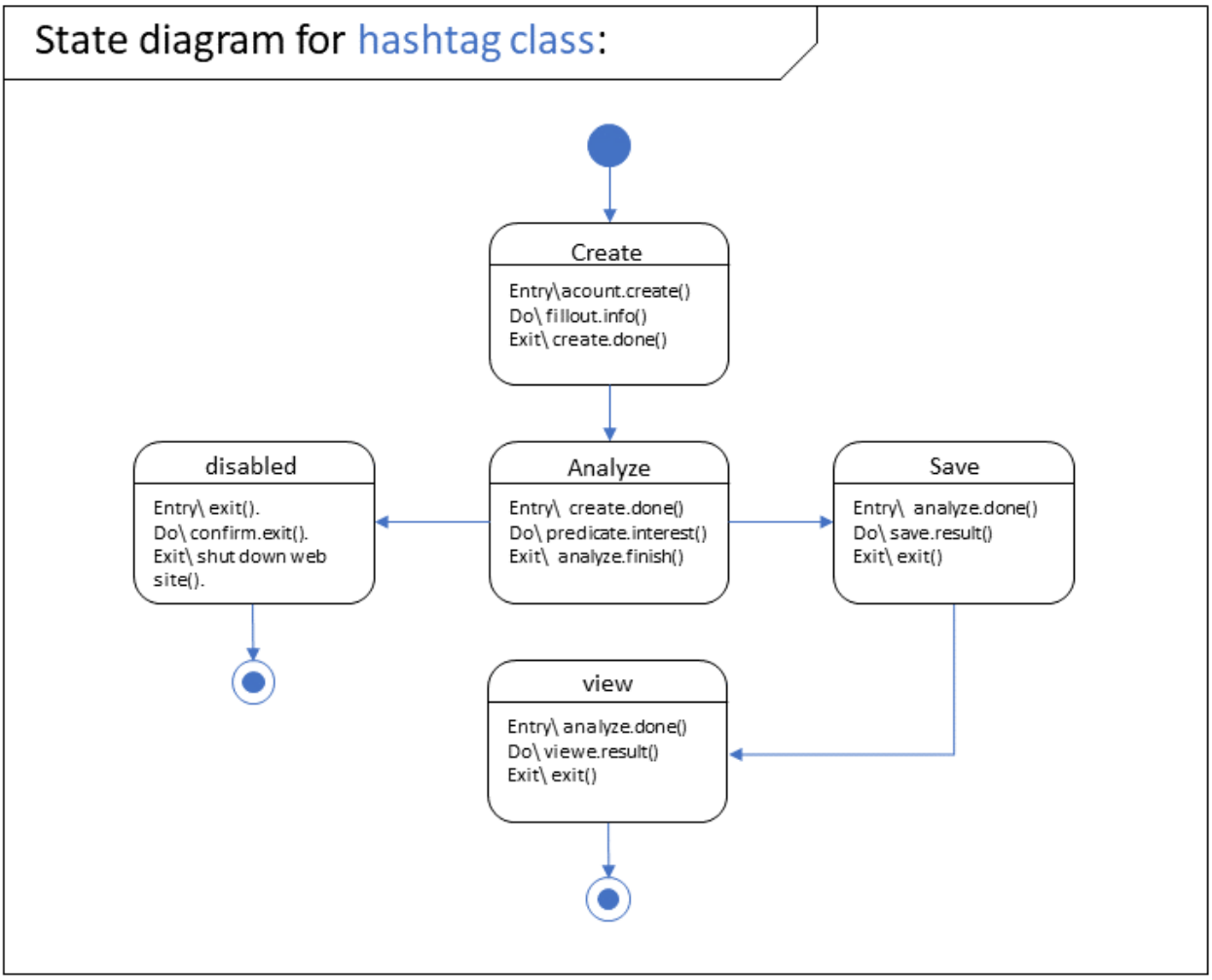
State diagram for analyzed account class:



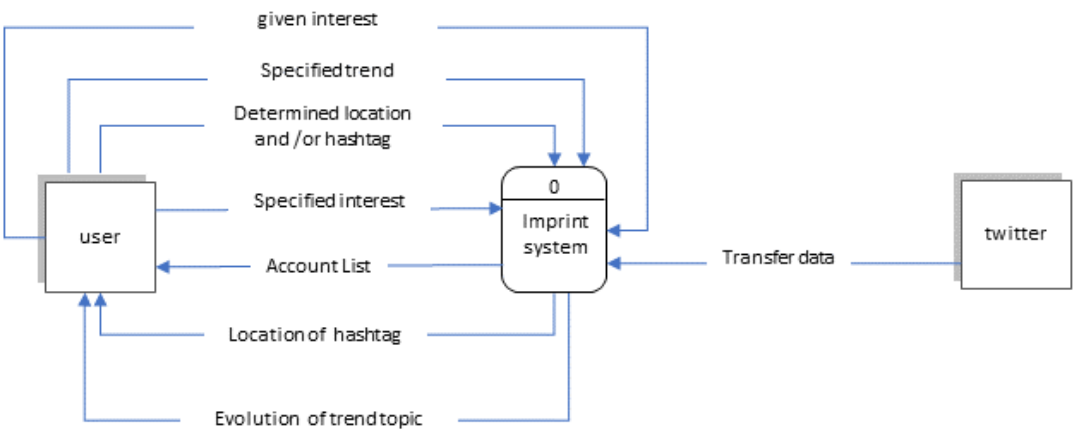
State diagram for trend class:



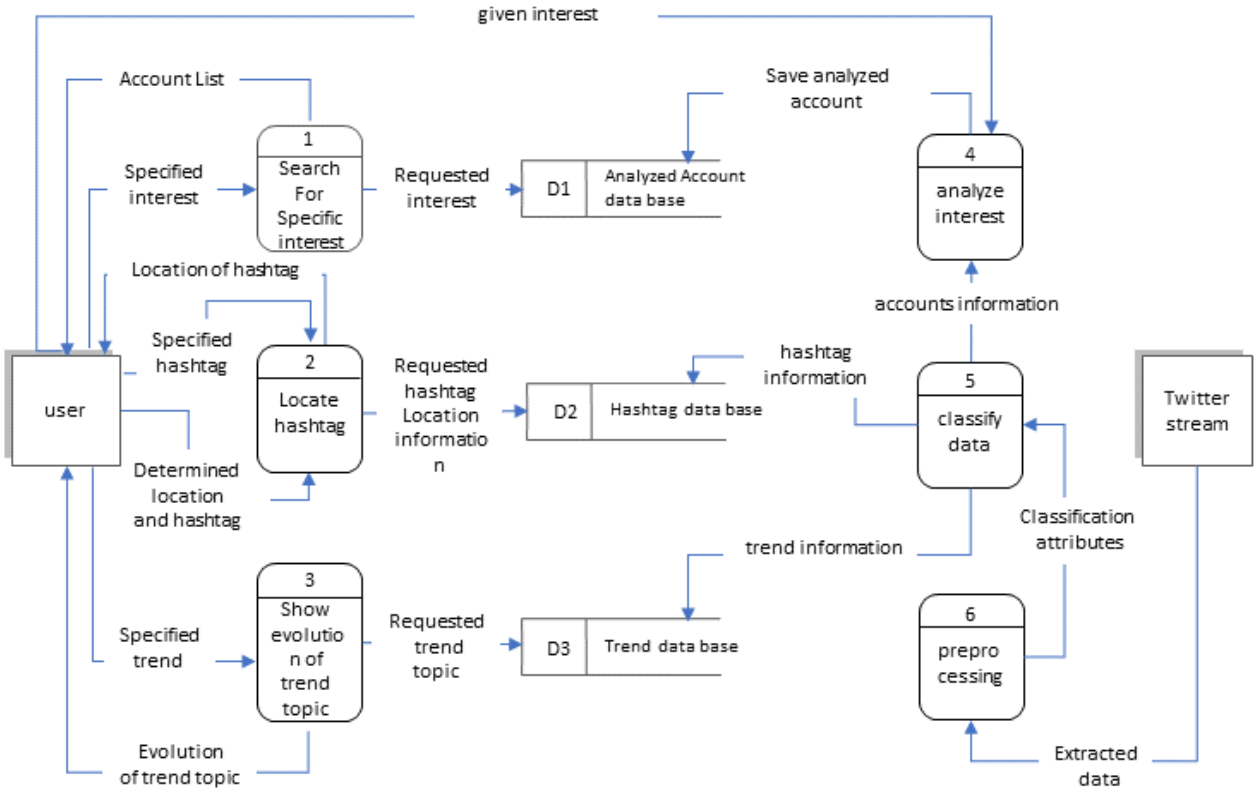
State diagram for hashtag class:



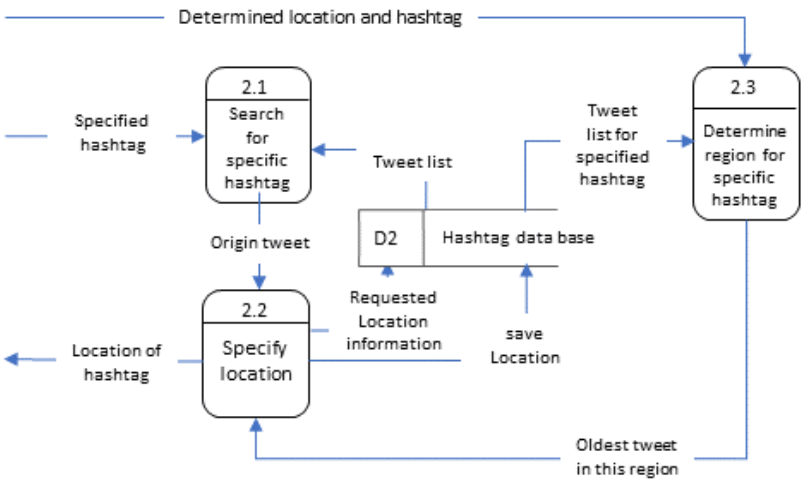
3.3 Data Flow Diagrams System Models



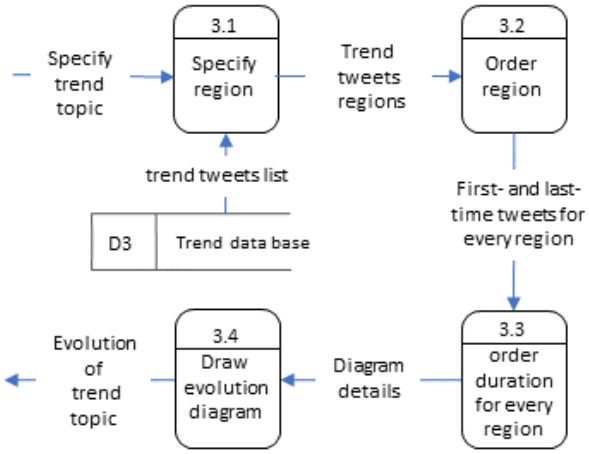
DFD: context diagram



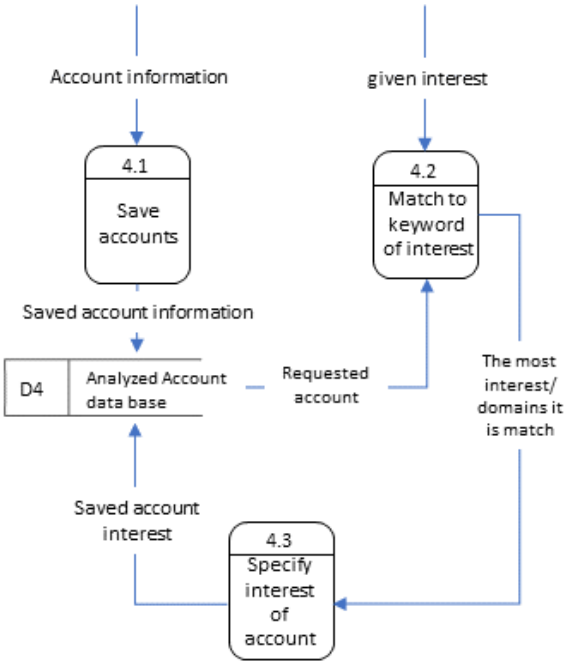
DFD: Level 0



DFD: Level 1 for process 2

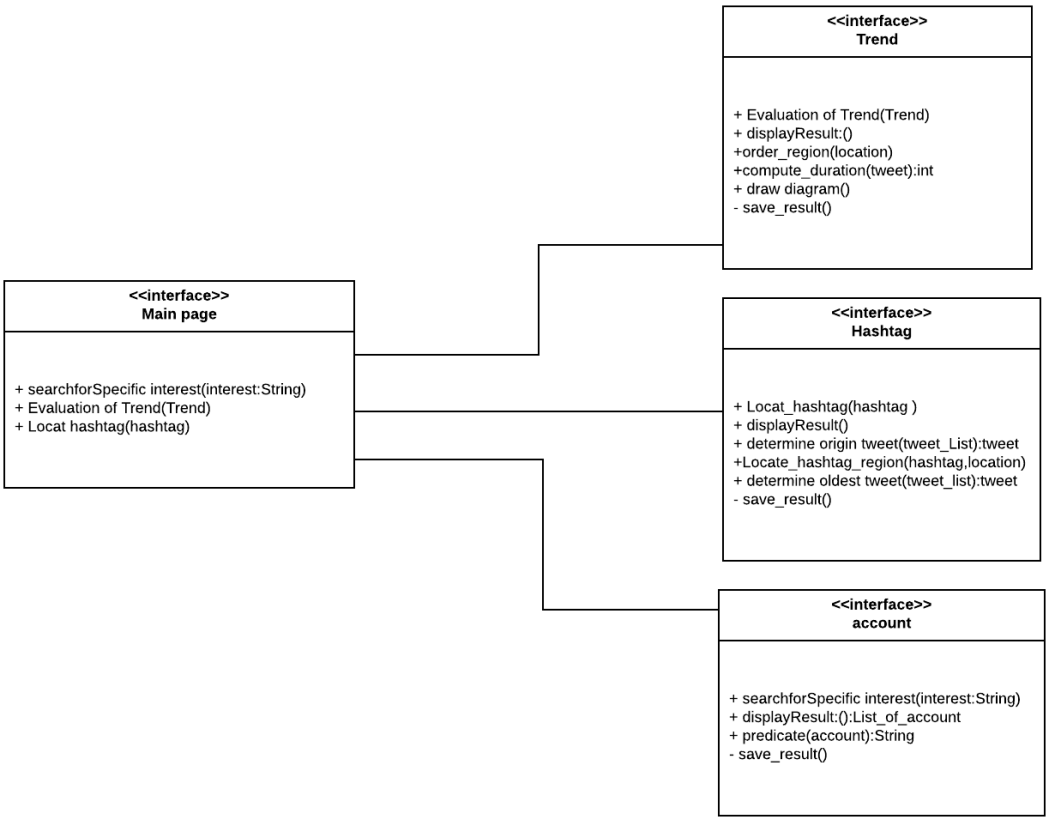


DFD: Level 1 for process 3

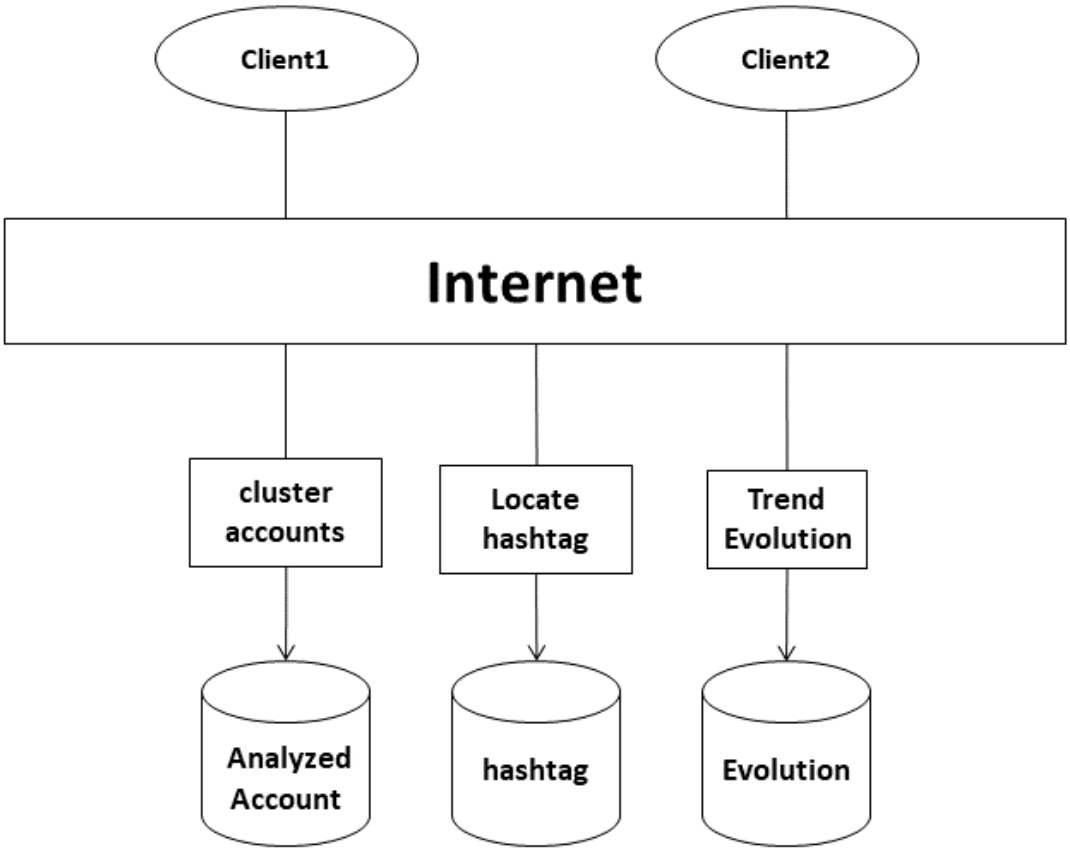


DFD: Level 1 for process 4

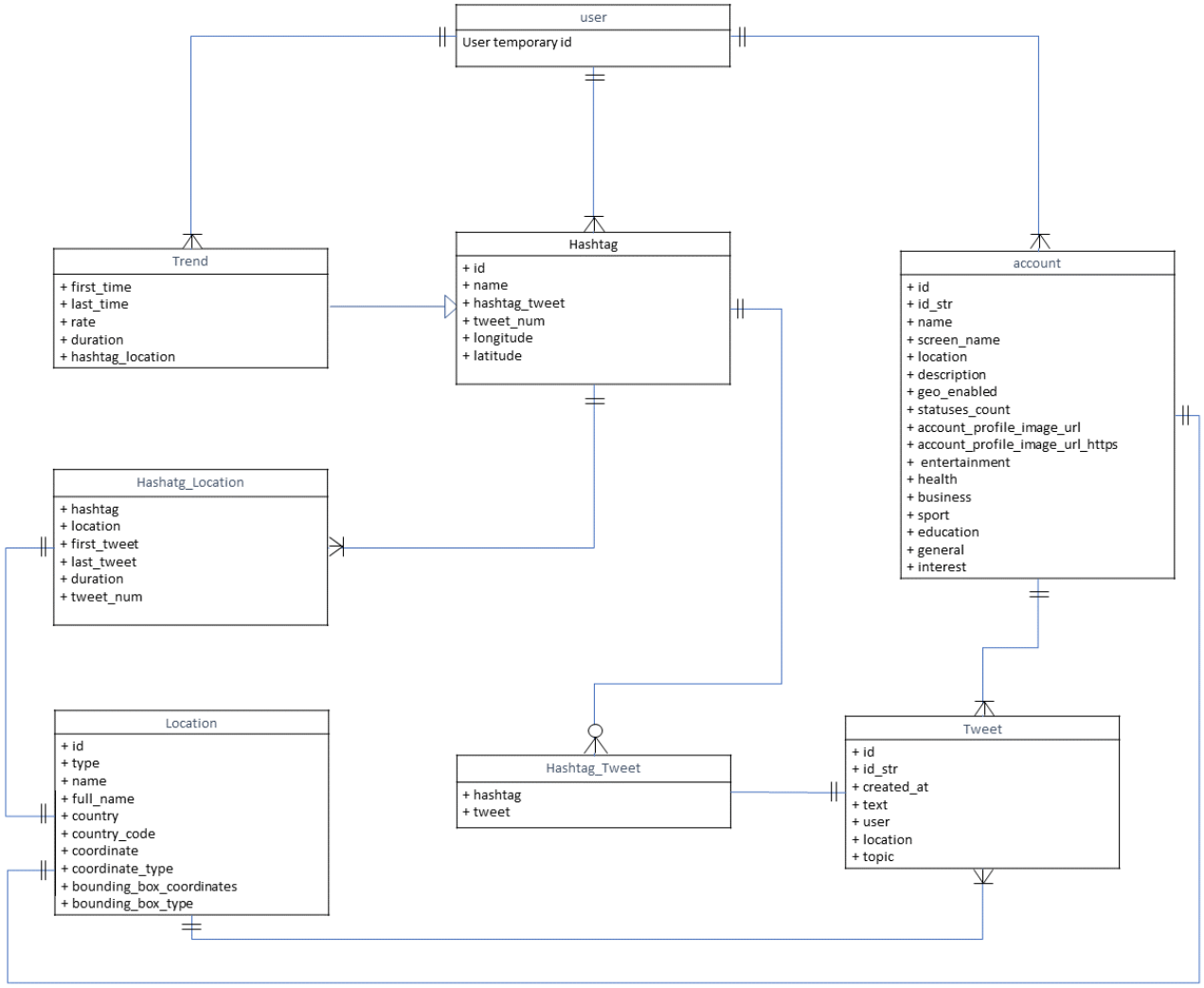
3.4 Interface Design



3.5 Architecture Design



3.6 ER Diagram



Summary:

This chapter introduces with functional and nonfunctional requirements, then we describe the proposed and alternative solutions, followed by UML system models, data flow diagrams, interface design, architecture design, and ER Diagram.

Chapter 4 SYSTEM DESIGN

4.1 Design Constraints

4.1.1 Hardware and Software Environment

Hardware:

Ram 16 GB**.** **Hard Disk** 64 GB.

Processor Intel(R). **Speed** 2.21 GHz.

Software:

Development:

-python -Postgres DB - php

Run-time:

Operating System windows 10.

4.1.2 End User Characteristics

users of the proposed system can search for: specific hashtag, or specific trend, list of account that fit the search query.

4.2 Architectural Strategies

4.2.1 Algorithm to Be Used

Search in Data Base:

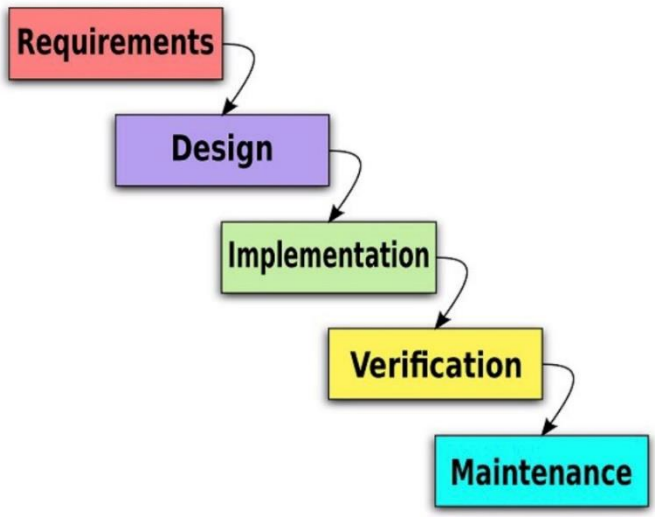
Using PostGIS an extension of Postgres DB means we are using R-tree algorithm to insert, search the location of tweets and users. Also, we are going to use Inverted Index to search for trending keywords and hashtags. R-tree has many advantages: R-tree design suitable for indexing multidimensional information like geographic, adds support for geographic objects, update data much faster to their database.

Text classification LDA Cluster:

In this study, we focus on Twitter data and build models for classifying “tweets” into user interests which is: general, education, health, sport entertainment, business using LDA cluster model is a completely unsupervised algorithm that models each document as a mixture of topics. by defining a one-to-one correspondence between LDA’s latent topics and user tags. This allows Labeled LDA to directly learn word-tag correspondences LDA works by first making a key assumption: the way a document was generated was by picking a set of topics and then for each topic picking a set of words The model generates automatic summaries of topics in terms of a discrete probability distribution over words for each topic, and further infers per-document discrete distributions over topics. Most importantly, LDA makes the explicit assumption that each word is generated from one underlying topic. Although LDA is expressive enough to model.

4.2.2 Development Method

We will define the SDLC method that we are following We depend on the waterfall model which appear in the following figure:



Why we use waterfall model? to enhance the understanding of our project, in this model phases explain procedures and completed one at a time, and phases do not overlap.

Phase 1: System and Software Requirements

The planning phase is the startup of the project where define the need of the proposed system. to meet the project's objectives the idea of our project is examined carefully by many aspects like the ability of our knowledge, our limited resources that we have.

Phase 2: System Analysis phase

this phase includes gather information about the proposed system after that identified the end user gathered requirements to stand on functional and non-functional requirements which led us to develop the system proposal.

Phase 3: Design phase

In this phase, we determine the basic architecture design of the system which describes the hardware, software environments in our system that we will use.

Phase 4: Implementation phase

During the final phase, the project is built and put into execution. Finally, test all the work done to check if it meets the requirement.

Phase 5: Testing phase

In this phase, it concerns about verification test to check any error and validation test to validate if the system meets the user requirements.

4.2.3 Future Enhancements/Plans

- implement these methods on Arabic tweets.
- Keep logs of users' search queries.
- make sure the tweets are related to the hashtag and topic and filter the noise data.
- Show the most common hashtags that are related to the particular topic.
- level up the prediction functions to include the followers and friend list and build/ draw diagrams and maps to show the connections.
- Predict user location and gender by predicate it from their tweets.

4.3 Implementation Languages

using python, its more flexible with twitter and it have Many Open Source Frameworks and Tools, and also it has large and robust standard library makes Python score over other programming languages.

Summary:

This chapter concerns about design consideration, first explain about the environment of hardware and software we will implement the system with, And list user characteristics. Then explain the algorithm

used, reuse of component, explain waterfall development method and its phases, list future plans and last type about python programming language that system will be used.

Chapter 5 Implementation and Validation

5.1 Implementation:

5.1.1 python Implementation:

5.1.1.1 imprint

```
# plugins need to execute this code
import pandas as pd


def read_json(file_name):
    ''' this method for load data from json file to pandas dataframe
    1- select all tweet written in English
    2- select important information which iclude when tweet created, its id, its content(text),
        its author(user),the entiteis includ in it like(hastag name, media, other),
        its place which represent name and other important information
        that help imprint system. finally longitutde and latitude of that tweet.
    3- send the resulted dataframe to columns_extractor method.
    '''
    df_original = pd.read_json(file_name, lines=True)
    df_en_tweets = df_original[ df_original.lang == "en"]
    df_en_tweets = df_en_tweets[["created_at", "id", "id_str", "text", 'user', 'entities',
                                 'place', 'coordinates']]
    columns_extractor(df_en_tweets)

```

5.1.1.2 converter

```
# the main processes
import converter
import filterData
import textClassification
import instancesCreation

# 1- call json file to begin process.
json_file_name = 'part1.json'
# 2- this class is charge for convert from json extension to CSV extension.
converter.read_json(json_file_name)
# 3- the result of converter which will saved in csv file called data pass to filter data code
#  by assign file namt to the filter data global variable,
#  it is take care of all information and clean it, and save to new csv file.
converted_file = 'data.csv'
filterData.load_data(converted_file)
# 4- text classification is take result of filter data and used for extract features from text
# and do some analysis and save it to new csv file called classifier.
textClassification.file_name = 'filtered data.csv'
# 5- final step, the result of filter and text classification code pass to create instances code to
# make the instances of each class in the system "the define of each class present in classes code"
instancesCreation.main()

```

```
def columns_extractor(df_en_tweets):
    ''' this method responsible of extract attribute for each important column then append them with
    the passed data frame which has english tweet just, then call selected data method '''
    user_col = df_en_tweets['user'].apply(pd.Series).add_prefix('user__')
    entities_col = df_en_tweets['entities'].apply(pd.Series).add_prefix('entities__')
    place_col = df_en_tweets['place'].apply(pd.Series).add_prefix('place__')
    coordinates_col = df_en_tweets['coordinates'].apply(pd.Series).add_prefix('coordinates__')
    df_en_tweets_col = df_en_tweets.columns.difference(['user','entities', 'place', 'coordinates'])
    df_extracted = pd.concat([df_en_tweets[df_en_tweets_col],user_col, entities_col, place_col, coordinates_col], axis=1)
    select_data(df_extracted)


def select_data(df_extracted):
    ''' this method care of selcting desierd data with extract attribute then again append it to the original one'''
    df_selected = df_extracted[['created_at', 'id', 'id_str', 'text', 'user__id', 'user__id_str',
            'user__statuses_count','user__name', 'user__screen_name', 'user__location','user__geo_enabled',
                        'user__profile_image_url', 'user__profile_image_url_https',
            'entities__hashtags','place__country','place__country_code', 'place__full_name', 'place__id','place__name',
            'place__place_type', 'coordinates__coordinates','coordinates__type']]
    entities__hashtags_text_col = df_extracted['entities__hashtags'].apply(lambda x: [d['text'] for d in x])
    coordinates__coordinates_col = df_extracted['coordinates__coordinates'].apply(pd.Series).add_prefix('coordinates__coordinates__')
    df_selected_col = df_selected.columns.difference(['entities__hashtags'])
    df_convereted = pd.concat([df_selected[df_selected_col], entities__hashtags_text_col, coordinates__coordinates_col], axis=1)
    save_to_csv(df_convereted)
```

```
def save_to_csv(df_convereted):
    ''' this method save data frame to csv file'''
    df_convereted.to_csv('data.csv', index=False)
```

5.1.1.3 filter Data

```
# plugins need to execute this code
from langdetect.lang_detect_exception import LangDetectException
from nltk import WordPunctTokenizer
from langdetect import detect
import preprocessor as p
import pandas as pd
import re
import time


def load_data(file_name):
    ''' this method load csv file todata frame '''
    df_converted = pd.read_csv(file_name)
    filter_data(df_converted)
    return df_converted
```

```python

def filter_data(df_converted):
    # assign same number of columns of data frame 1 to data frame 2 which is empty
    # 1- clean tweet text
    df_cleaned_text = clean_tweet_text_col(df_converted)
    # 2- drop null tweet text
    df_cleaned_text = df_cleaned_text[~df_cleaned_text.text.isnull()]
    # 3- detect tweet lang
    df_detected_text_lang = detect_tweet_text_lang(df_cleaned_text)
    # 4- keep english tweet
    df_en_text = df_detected_text_lang[df_detected_text_lang.lang == 'en'].set_index("created_at").reset_index()
    # 5- drop where geo_enabled is false
    df_filtered = df_en_text[df_en_text.user__geo_enabled == True]
    df_filtered = df_filtered[~df_en_text.coordinates__coordinates__0.isnull()]
    # 6- save it to csv
    df_filtered.to_csv('filtered data.csv', index=False)


def check_null(text):
    # return boolean value upon discover nulls.
    if str(text) == '' or str(text) == 'nan' or not str(text) or len(str(text)) == 0:
        return True
    else:
        return False


def clean_tweets(tweet):
    ''' it takes 1 tweet text:
        1- check null and assign it to nan value.
        2- after tweepy preprocessing the colon left remain after removing mentions or RT sign in the
         beginning of the tweet replace consecutive non-ASCII characters with a space, links removed, lower case
         is applied define WordPunctTokenizer() instances then append the result.
    '''
    if check_null(tweet):
        clean_tweet = 'nan'
    else:
        tweet = re.sub(r':', '', tweet)
        tweet = re.sub(r',Ä¶', '', tweet)
        tweet = re.sub(r'[^\x00-\x7F]+', ' ', tweet)
        user_removed = re.sub(r'@[A-Za-z0-9]+', '', tweet)
        link_removed = re.sub('https?://[A-Za-z0-9./]+', '', user_removed)
        number_removed = re.sub('[^a-zA-Z]', ' ', link_removed)
        lower_case_tweet = number_removed.lower()
        tok = WordPunctTokenizer()
        words = tok.tokenize(lower_case_tweet)
        clean_tweet = (' '.join(words)).strip()
    return clean_tweet


def clean_tweet_text_col(df_converted):
    ''' this method clean text column using a "preprocessor as p" tweepy preprocessing
        then send the result to "clean tweet" method
    '''
    # for loop to go through rows to detect language for text len(df1) stand for number of rows on the data frame 2
    for i in range(len(df_converted)):
        # clean text and description fields
        filtered_text = clean_tweets(p.clean(df_converted.at[i, 'text']))
        # assign the cleaned new value to the data frame 2
        df_converted._set_value([i], ['text'], filtered_text)
    return df_converted

```

```
77
78
79  def detect_tweet_text_lang(df_cleaned_text):
80      ''' this method care of the tweet has been selected and cleaned but it is not contain an english text
81          using "langdetect "plugins.
82          1- add new colun to assign detected language.
83          2- using "for loop" to care of all tweets.
84          3- using handle to make a language filter "try and except" a null, numbers,
85           url links text will raise an error and make detect raise an error.
86          4- in except the cleaned data frame will drop these tweets.
87      '''
88      df_cleaned_text['lang'] = ''
89      # for loop to go through rows to detect language for text
90      for i in range(len(df_cleaned_text)):
91          try:
92              d = detect(df_cleaned_text.at[i, 'text'])
93              df_cleaned_text._set_value([i], ['lang'], d)
94          except LangDetectException:
95              df_cleaned_text.drop(df_cleaned_text.index[i])
96      return df_cleaned_text
97
```

5.1.1.4 text classification

```
1   # plugins need to execute this code
2   from sklearn.decomposition import LatentDirichletAllocation
3   from sklearn.feature_extraction.text import CountVectorizer
4   from nltk.stem import WordNetLemmatizer
5   from nltk.corpus import stopwords
6   import pandas as pd
7   import numpy as np
8
9
10  global file_name
11
12
13  def load_data(file_name):
14      ''' this method load csv file todata frame '''
15      df_filtered_data = pd.read_csv(file_name)
16      preprocessing(df_filtered_data)
17
```

```
def preprocessing(df_filtered_data):
    wordnet_lemmatizer = WordNetLemmatizer()
    nrows = len(df_filtered_data)
    lemmatized_text_list = []
    for row in range(0, nrows):
        # Create an empty list containing lemmatized words
        lemmatized_list = []  # return word to origin...
        # Save the text and its words into an object
        text = df_filtered_data.at[row, 'text']
        text_words = text.split(" ")
        # Iterate through every word to lemmatize
        for word in text_words:
            lemmatized_list.append(wordnet_lemmatizer.lemmatize(word, pos="v"))
        # Join the list
        lemmatized_text = " ".join(lemmatized_list)
        # Append to the list containing the texts
        lemmatized_text_list.append(lemmatized_text)
    df_filtered_data['text'] = lemmatized_text_list
    # Loading the stop words in english
    stop_words = list(stopwords.words('english'))
    df_filtered_data['text'] = df_filtered_data['text']
    for stop_word in stop_words:  # remove stop word
        regex_stopword = r"\b" + stop_word + r"\b"
        df_filtered_data['text'] = df_filtered_data['text'].str.replace(regex_stopword, '')
    # clean text from any spaces left
    df_filtered_data['text'] = df_filtered_data['text'].apply(lambda x: ' '.join([w for w in x.split() if len(w) > 3]))
    df_filtered_data['text'] = df_filtered_data['text'].str.replace("\r", " ")
    df_filtered_data['text'] = df_filtered_data['text'].str.replace("\n", " ")
    df_filtered_data['text'] = df_filtered_data['text'].str.replace("    ", " ")
    # pass it to creat_lda_model method
    creat_lda_model(df_filtered_data)
    return df_filtered_data


def label_theme(row):
    ''' this method label theme with the resulted from dominant_topic number LDA module extract features'''
    if row['dominant_topic'] == 1 or row['dominant_topic'] == 2 \
            or row['dominant_topic'] == 4 or row['dominant_topic'] == 7 :
        return 'entertainment'
    if row['dominant_topic'] == 9:
        return 'health'
    if row['dominant_topic'] == 0 or row['dominant_topic'] == 3 \
            or row['dominant_topic'] == 5 or row['dominant_topic'] == 10 or row['dominant_topic'] == 13:
        return 'General'
    if row['dominant_topic'] == 8 or row['dominant_topic'] == 14:
        return 'business'
    if row['dominant_topic'] == 6 or row['dominant_topic'] == 12:
        return 'sport'
    if row['dominant_topic'] == 11:
        return 'Education'


def show_topics(vectorizer, lda_model, n_words):
    ''' list word data frame will used in LDA module '''
    keywords = np.array(vectorizer.get_feature_names())
    topic_keywords = []
    for topic_weights in lda_model.components_:
        top_keyword_locs = (-topic_weights).argsort()[:n_words]
        topic_keywords.append(keywords.take(top_keyword_locs))
    return topic_keywords

```

```
80
81
82  def creat_lda_model(df_filtered_data):
83      # create CountVectorizer instances to extract the words needed
84      vectorizer = CountVectorizer(analyzer='word', max_df=0.5, min_df=2, stop_words='english', lowercase=True,
85                                   token_pattern='[a-zA-Z0-9]{3,}', max_features=5000)
86
87      data_vectorized = vectorizer.fit_transform(df_filtered_data['text'])
88      # create LDA model by pass number of topic
89      lda_model = LatentDirichletAllocation(n_components=15,  # Number of topics
90                                            learning_method='online', random_state=0, n_jobs=-1  # Use all available CPUs
91                                            )
92      lda_output = lda_model.fit_transform(data_vectorized)
93      # save word data frame
94      topic_keywords = show_topics(vectorizer=vectorizer, lda_model=lda_model, n_words=16)
95      # assign keywords to new dataframe , define cols name word ,rows name topic
96      df_topic_keywords = pd.DataFrame(topic_keywords)
97      df_topic_keywords.columns = ['Word ' + str(i) for i in range(df_topic_keywords.shape[1])]
98      df_topic_keywords.index = ['Topic ' + str(i) for i in range(df_topic_keywords.shape[0])]
99      df_topic_keywords.to_csv('word.csv', index=False)
```

```
100     # rename topics
101     Topics_theme = ['theme1', 'theme2', 'theme3', 'theme4', 'theme5',
102                     'theme6', 'theme7', 'theme8', 'theme9', 'theme10',
103                     'theme11', 'theme12', 'theme13', 'theme14', 'theme15']
104     # add new column to assign Topics_theme
105     df_topic_keywords['topic_theme'] = Topics_theme
106     df_topic_keywords.set_index('topic_theme', inplace=True)
107     # Create Document - Topic Matrix to classify the words
108     lda_output = lda_model.transform(data_vectorized)
109     # column names Reflect the DataFrame
110     topicnames = df_topic_keywords.T.columns
111     # index names
112     docnames = ["Doc" + str(i) for i in range(len(df_filtered_data))]
113     # Make the pandas dataframe
114     df_document_topic = pd.DataFrame(np.round(lda_output, 2), columns=topicnames, index=docnames)
115     # Get dominant topic for each document
116     dominant_topic = np.argmax(df_document_topic.values, axis=1)
117     # add new column to assign dominant_topic number
118     df_document_topic['dominant_topic'] = dominant_topic
119     df_document_topic.reset_index(inplace=True)
120     # send result to merge_result function
121     merge_result(df_document_topic)
122     return df_document_topic
123
```

```
125 def merge_result(df_document_topic):
126     df_filtered_data = pd.read_csv(file_name)
127     # append new columns to original dataframe
128     df_filtered_data = pd.merge(df_filtered_data, df_document_topic['dominant_topic'], left_index=True, right_index=True)
129     # convert numbers to topic name
130     df_filtered_data['dominant_topic_theme'] = df_filtered_data.apply(lambda row: label_theme(row), axis=1)
131     df_filtered_data.to_csv('classifier.csv')
```

5.1.1.5 instances creation

```python
# plugins need to execute this code
from classes import Hashtag, Trend, Place, Hashtag_Place, User, Tweet, Hashtag_Tweet
from wordcloud import WordCloud
from itertools import chain
import pandas as pd


def chainer(s):
    # spilt text array to its content
    return list(chain.from_iterable(s.str.split(',')))


def convert_to_wordcloud(df_extracted_hashtags_text):
    long_string = ','.join(list(df_extracted_hashtags_text['entities__hashtags'].values))
    # Create a WordCloud object
    wordcloud = WordCloud(background_color="white", max_words=5000, contour_width=3, contour_color='steelblue')
    # Generate a word cloud
    wordcloud.generate(long_string)
    # Visualize the word cloud
    image = wordcloud.to_image()
    wordcloud.to_file('C:\AppServ\www\imprint\Hashatgs.png')
    # image.show()


def extract_hashtags_text(df_filtered_data):
    #  filter hashtag text from any [,] and' characters
    df_extracted_hashtags_text = pd.DataFrame({'entities__hashtags': chainer(
        df_filtered_data['entities__hashtags'].replace('\]', '', regex=True).replace('\[', '', regex=True).
            replace('\'', '', regex=True).astype(str).replace(" ", "", regex=True))}).sort_values(
        by=['entities__hashtags'])
    # order and display duplicate
    df_extracted_hashtags_text = df_extracted_hashtags_text[
        df_extracted_hashtags_text.duplicated(['entities__hashtags'], keep=False)].sort_values(
        by=['entities__hashtags'])
    # count duplicate and assign them to a new column which is sum
    df_extracted_hashtags_text['count'] = df_extracted_hashtags_text.groupby('entities__hashtags').cumcount()
    # drop duplicates rows and keep the last occurrence
    df_extracted_hashtags_text = df_extracted_hashtags_text.drop_duplicates('entities__hashtags', keep='last',
                                                                            ignore_index=True). \
        sort_values(by=['count'], ascending=False).set_index('entities__hashtags').reset_index()
    #  save result to csv to reused by convert_to_wordcloud method
    df_extracted_hashtags_text.to_csv('hashtags text.csv')
    convert_to_wordcloud(df_extracted_hashtags_text)
    return df_extracted_hashtags_text

```

```
46
47  def place_ceration(df_filtered_data_place):
48      # drop duplicate to avoid error in data base
49      df_filtered_data_place = df_filtered_data_place.drop_duplicates('place__id', keep='first').reset_index(drop=True)
50      # for loop for assign user instances and save them in data base
51      for i in range(len(df_filtered_data_place)):
52          placeId = df_filtered_data_place.at[i, 'place__id']
53          typeplace = df_filtered_data_place.at[i, 'place__place_type']
54          placeName = df_filtered_data_place.at[i, 'place__name']
55          fullName = df_filtered_data_place.at[i, 'place__full_name']
56          country_name = df_filtered_data_place.at[i, 'place__country']
57          countryCode = df_filtered_data_place.at[i, 'place__country_code']
58          coordinates_coordinates_0 = df_filtered_data_place.at[i, 'coordinates__coordinates__0']
59          coordinates_coordinates_1 = df_filtered_data_place.at[i, 'coordinates__coordinates__1']
60          coordinatesType = df_filtered_data_place.at[i, 'coordinates__type']
61          place = Place(placeId, typeplace, placeName, fullName, country_name, countryCode,
62                        coordinates_coordinates_0, coordinates_coordinates_1, coordinatesType)
63          place.save_result()
64
65
66  def trend_creation(hashtag, df_tweets_contain_hashtag):
67      # take last tweet assign it to end time
68      end_time = df_tweets_contain_hashtag.created_at.iloc[-1]
69      # compute duration
70      trend_duration = (end_time-hashtag.origin_tweet_time).total_seconds()
71      # compute rate
72      rate = hashtag.hashtag_tweet_num / trend_duration
73      # create trend instance and save it to data base
74      trend = Trend(hashtag.hashtag_id, hashtag.hashtag_name, end_time, hashtag.hashtag_tweet_num,
75                    hashtag.origin_tweet_time, hashtag.longitude, hashtag.latitude,trend_duration, rate)
76      trend.save_result()
```

```
80
81  def hashtag_place_creation(hashtag, place_id, df_tweets_contain_hashtag):
82      # make a data frame contain passes hastag id and place id and reset index
83      df_tweets_contain_hashtag_place = df_tweets_contain_hashtag[
84          df_tweets_contain_hashtag.place__id == place_id].sort_values(by=['created_at']).reset_index(drop=True)
85      # save length of data frame
86      hashtag_place_tweet_num = len(df_tweets_contain_hashtag_place)
87      first_tweet_time = df_tweets_contain_hashtag_place.iloc[0].created_at
88      # some handle of first and last time values
89      if len(df_tweets_contain_hashtag_place) == 1:
90          last_tweet_time = first_tweet_time
91      elif len(df_tweets_contain_hashtag_place) == 2:
92          last_tweet_time = df_tweets_contain_hashtag_place.at[1, 'created_at']
93      else:
94          # take last twwet time which present in last of dataframe
95          last_tweet_time = df_tweets_contain_hashtag_place.created_at.iloc[-1]
96      #     compute duration
97      duration = (last_tweet_time - first_tweet_time).total_seconds()
98      # craet hashtag_place instances
99      hashtag_place = Hashtag_Place(hashtag.hashtag_id, place_id, first_tweet_time, last_tweet_time,
100                                    hashtag_place_tweet_num,duration)
101     hashtag_place.save_result()
102
103
104 def hashtag_tweet_saving(df_tweets_contain_hashtag_tweet, hashtag, df_tweets_contain_hashtag):
105     '''this method for saving relation ship between tweet and hastag in csv file to call it later after tweet instances called'''
106     df_temp = df_tweets_contain_hashtag[['id']]
107     df_temp.insert(0, 'hashtag_id', hashtag.hashtag_id)
108     df_tweets_contain_hashtag_tweet = df_tweets_contain_hashtag_tweet.append(df_temp).reset_index(drop=True)
109     return df_tweets_contain_hashtag_tweet
```

```python
def hashtag_tweet_creation():
    ''' the file save realtion between hashtag and tweet called to used for unsert hashtag_tweet then save it'''
    df_tweets_contain_hashtag_tweet = pd.read_csv('hashtag_tweet.csv')
    for i in range(len(df_tweets_contain_hashtag_tweet)):
        hashtag_id = df_tweets_contain_hashtag_tweet.at[i,'hashtag_id']
        tweet_id = df_tweets_contain_hashtag_tweet[i,'id']
        hashtag_tweet = Hashtag_Tweet(hashtag_id,tweet_id)
        hashtag_tweet.save_result()


def tweet_creation(df_tweet):
    ''' tweet instances creation '''
    df_tweet = df_tweet[
        ['created_at', 'id', 'id_str', 'text', 'user__id', 'place__id', 'dominant_topic_theme']]
    for i in range(len(df_tweet)):
        created = df_tweet.at[i, 'created_at']
        id_tweet = df_tweet.at[i, 'id']
        id_str = df_tweet.at[i, 'id_str']
        text = df_tweet.at[i, 'text']
        user_id = df_tweet.at[i, 'user__id']
        place_id = df_tweet.at[i, 'place__id']
        topic = df_tweet.at[i, 'dominant_topic_theme']
        tweet = Tweet(created, id_tweet, id_str, text, user_id, place_id, topic)
        tweet.save_result()
    return 0
```

```python
def user_creation(df_users):
    #  compute tweets by single user
    for i in range(len(df_users)):
        #  all values needed by user instances
        user__id = df_users.at[i, 'user__id']
        user__geo_enabled = df_users.at[i, 'user__geo_enabled']
        user__id_str = df_users.at[i, 'user__id_str']
        user__location = df_users.at[i, 'user__location']
        user__name = df_users.at[i, 'user__name']
        user__profile_image_url = df_users.at[i, 'user__profile_image_url']
        user__profile_image_url_https = df_users.at[i, 'user__profile_image_url_https']
        user__screen_name = df_users.at[i, 'user__screen_name']
        user__statuses_count = df_users.at[i, 'user__statuses_count']
        # assign interest by computing : number of tweet
        df_user = df_users[df_users.user__id == user__id]
        df_user = df_user[['user__id', 'dominant_topic_theme']].set_index('user__id').reset_index(drop=True)
        user_tweet_num = len(df_user)
        # interest can not specify by 1 tweets so assign interest as undefiend
```

```
        # interest can not specify by 1 tweets so assign interest as undefiend
        if user_tweet_num < 2:
            interest = 'undefiend'
            entertainment = 0
            health = 0
            business = 0
            sport = 0
            education = 0
            general = 0
        else:
            entertainment = 0
            health = 0
            business = 0
            sport = 0
            education = 0
            general = 0
            # compute number of tweet in every topic
            df_user = df_user.groupby(['dominant_topic_theme']).dominant_topic_theme.agg(
                count_col=pd.NamedAgg(column="dominant_topic_theme", aggfunc="count"))
            df_user.to_csv('count_interest.csv')
            df_user = pd.read_csv('count_interest.csv')
            #  compute percent by this method
            df_user['persent'] = (df_user.count_col / user_tweet_num) * 100
            #  label max value with its interest
            interest = df_user.loc[df_user['persent'].idxmax()].dominant_topic_theme
        # user creation
        user = User(user__id, user__id_str, user__name, user__screen_name, user__location,
                    user__geo_enabled, user__statuses_count, user__profile_image_url, user__profile_image_url_https,
                    user_tweet_num, entertainment, health, business, sport, education, general, interest, df_user['persent'])
        user.save_result()
```

```
def classes_creation(df_filtered_data, df_extracted_hashtags_text):
    # creation hashtag instances
    #  initiate for data frame have all hashtag id and tweet id to append it later with other result
    df_tweets_contain_hashtag_tweet = pd.DataFrame(columns=['id', 'hashtag_id'])
    # use hashtag name to search for tweet
    for i in range(len(df_extracted_hashtags_text)):
        hashtag_id = i
        hashtag_name = "'" + df_extracted_hashtags_text.at[i, 'entities__hashtags'] + "'"  # hashtag_name
        df_tweets_contain_hashtag = df_filtered_data[
            df_filtered_data.entities__hashtags.str.contains(hashtag_name, regex=[False])].sort_values(
            by=['created_at']).reset_index(drop=True)
        temp = df_filtered_data.entities__hashtags.iloc[i]
        # assign length of data frame resulted to hastag tweet number
        hashtag_tweet_num = len(df_tweets_contain_hashtag)  # 0 job  tweetnum origin coordinate0 coordinate 1
        # First tweet id --> place coordinate long lat
        origin_tweet_time = df_tweets_contain_hashtag.iloc[0].created_at
        longitude = df_tweets_contain_hashtag.iloc[0].coordinates__coordinates__0
        latitude = df_tweets_contain_hashtag.iloc[0].coordinates__coordinates__1
        hashtag = Hashtag(hashtag_id, hashtag_name, hashtag_tweet_num, origin_tweet_time, longitude, latitude)
        hashtag.save_result()
        # trend creation upon sort from max to lowest
        if i < 3:
            trend_creation(hashtag, df_tweets_contain_hashtag)
        # save the hashtag_tweet to csv file
        df_tweets_contain_hashtag_tweet = hashtag_tweet_saving(df_tweets_contain_hashtag_tweet, hashtag,
                                                               df_tweets_contain_hashtag)
        df_tweets_contain_hashtag_tweet.to_csv('hashtag_tweet.csv')
        # creation of place_hashtag instances
        for j in range(len(df_tweets_contain_hashtag)):
            place_id = df_tweets_contain_hashtag.at[j, 'place__id']
            hashtag_place_creation(hashtag, place_id, df_tweets_contain_hashtag)

```

5.1.1.6 classes

```python
# plugins need to execute this code
import psycopg2



class Place:
    def __init__(self, place_id, type, place_name, full_name, country, country_code, coordinates_coordinates_0,
                 coordinates_coordinates_1, coordinates_type):
        self.place_id = place_id
        self.type = type
        self.place_name = place_name
        self.full_name = full_name
        self.country = country
        self.country_code = country_code
        self.coordinates_coordinates_0 = coordinates_coordinates_0
        self.coordinates_coordinates_1 = coordinates_coordinates_1
        self.coordinates_type = coordinates_type

    def save_result(self):
        con = psycopg2.connect(database="postgres", user="postgres", password="1234", port="5432")
        print("Database opened successfully")
        cur = con.cursor()
        cur.execute(
            "INSERT INTO place (place_id, type,place_name, full_name, country, country_code, coordinates_coordinates_0,"
            " coordinates_coordinates_1,  coordinates_type) VALUES (%s,%s,%s,%s,%s,%s,%s,%s)", (
            self.place_id, type, self.place_name, self.full_name, self.country, self.country_code,
            self.coordinates_coordinates_0, self.coordinates_coordinates_1,
            self.coordinates_type))
        print("Operation done successfully")
        con.commit()
        con.close()

```

```python

class Hashtag:
    def __init__(self, hashtag_id, hashtag_name, hashtag_tweet_num, origin_tweet_time, longitude, latitude):
        self.hashtag_id = hashtag_id
        self.hashtag_name = hashtag_name
        self.hashtag_tweet_num = hashtag_tweet_num
        self.origin_tweet_time = origin_tweet_time
        self.longitude = longitude
        self.latitude = latitude

    def save_result(self):
        con = psycopg2.connect(database="postgres", user="postgres", password="1234", port="5432")
        print("Database opened successfully")
        cur = con.cursor()
        cur.execute(
            "INSERT INTO hashtag (hashtag_id, hashtag_name, hashtag_tweet_num, origin_tweet_time, longitude, latitude) "
            "VALUES VALUES (%s,%s,%s,%s,%s,%s)"
            , (self.hashtag_id, self.hashtag_name, self.hashtag_tweet_num, self.origin_tweet_time, self.longitude,
               self.latitude))
        print("Operation done successfully")
        con.commit()
        con.close()

```

```
56
57  class Tweet:
58      def __init__(self, tweet_id, id_str, text, created_at, user_id,
59                   place_id, topic):
60          self.tweet_id = tweet_id
61          self.tweet_id_str = id_str
62          self.created_at = created_at
63          self.text = text
64          self.user_id = user_id
65          self.place_id = place_id
66          self.topic = topic
67
68      def save_result(self):
69          con = psycopg2.connect(database="postgres", user="postgres", password="1234", port="5432")
70          print("Database opened successfully")
71          cur = con.cursor()
72          cur.execute("INSERT INTO tweet (tweet_id, id_str, text, created_at, user_id, place_id, topic) VALUES "
73                      "(%s, %s, %s, %s, %s, %s, %s)", (
74                          self.tweet_id, self.tweet_id_str, self.text, self.created_at, self.user_id, self.place_id,
75                          self.topic))
76          print("Operation done successfully")
77          con.commit()
78          con.close()
79
80
```

```
97  class Hashtag_Place:
98      def __init__(self, hashtag_id, place_id, first_tweet_time, last_tweet_time, hashtag_place_tweet_num, duration):
99          self.hashtag_id = hashtag_id
100         self.place_id = place_id
101         self.first_tweet_time = first_tweet_time
102         self.last_tweet_time = last_tweet_time
103         self.hashtag_place_duration = duration
104         self.hashtag_place_tweet_num = hashtag_place_tweet_num
105         self.duration = duration
106
107     def save_result(self):
108         con = psycopg2.connect(database="postgres", user="postgres", password="1234", port="5432")
109         print("Database opened successfully")
110         cur = con.cursor()
111         cur.execute(
112             "INSERT INTO hashtag_place (hashtag_id, place_id, first_tweet_time, last_tweet_time,"
113             "hashtag_place_duration, hashtag_place_tweet_num) VALUES "
114             "(%s, %s, %s, %s, %s, %s)", (
115                 self.hashtag_id, self.place_id, self.first_tweet_time, self.last_tweet_time,
116                 self.hashtag_place_duration,
117                 self.duration))
118         print("Operation done successfully")
119         con.commit()
120         con.close()
```

```
class User:
    def __init__(self, user_id, user_id_str, user_name, screen_name, location, user_geo_enabled, user_statuses_count
                 , user_profile_image_url, user_profile_image_url_https, user_tweet_num, entertainment,
                 health, business, sport, education, general, interest, persent):
        self.user_id = user_id
        self.user_id_str = user_id_str
        self.user_name = user_name
        self.screen_name = screen_name
        self.location = location
        self.user_geo_enabled = user_geo_enabled
        self.user_statuses_count = user_statuses_count
        self.user_profile_image_url = user_profile_image_url
        self.user_profile_image_url_https = user_profile_image_url_https
        self.user_tweet_num = user_tweet_num
        self.entertainment = entertainment
        self.health = health
        self.business = business
        self.sport = sport
        self.education = education
        self.general = general
        self.interest = interest
        self.persent = persent

    def save_result(self):
        con = psycopg2.connect(database="postgres", user="postgres", password="1234", port="5432")
        print("Database opened successfully")
        cur = con.cursor()
        cur.execute("INSERT INTO users (user_id, user_id_str, user_name, screen_name, location, user_geo_enabled,"
                    " user_statuses_count, user_profile_image_url, user_profile_image_url_https, user_tweet_num,"
                    " entertainment, health, business, sport, education, general, interest, persent) VALUES "
                    "( %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s)",
                    (self.user_id, self.user_id_str, self.user_name, self.screen_name,
                     self.location, self.user_geo_enabled, self.user_statuses_count, self.user_profile_image_url,
                     self.user_profile_image_url_https, self.user_tweet_num, self.entertainment, self.health,
                     self.business, self.sport, self.education, self.general, self.interest, self.persent))
        print("Operation done successfully")
        con.commit()
        con.close()


class Trend(Hashtag):  # inhertance from Hashtag
    def __init__(self, trend_id, trend_name, end_time, trend_tweet_num, origin_tweet_time, longitude, latitude,
                 trend_duration, rate):
        super().__init__(trend_id, trend_name, trend_tweet_num, origin_tweet_time, longitude, latitude)
        self.start_time = origin_tweet_time
        self.end_time = end_time
        self.trend_duration = trend_duration
        self.rate = rate

        con = psycopg2.connect(database="postgres", user="postgres", password="1234", port="5432")
        print("Database opened successfully")
        cur = con.cursor()
        cur.execute(
            "INSERT INTO trend (trend_id, trend_name, trend_tweet_num, rate, start_time, end_time, duration) VALUES "
            "(%s, %s, %s, %s, %s, %s, %s, %s,)", (
                self.hashtag_id, self.hashtag_name, self.hashtag_tweet_num, self.rate, self.start_time, self.end_time,
                self.trend_duration))
        print("Operation done successfully")
        con.commit()
        con.close()
```

5.1.2 php Implementation:

5.1.1.1 Home page

5.1.1.2 Search for interest

The system allow user to search for specific interest from the six domains which is general, entertainment, sport, education, health, business. The system will list the accounts with a percentage represent how much this account interest in this topic.

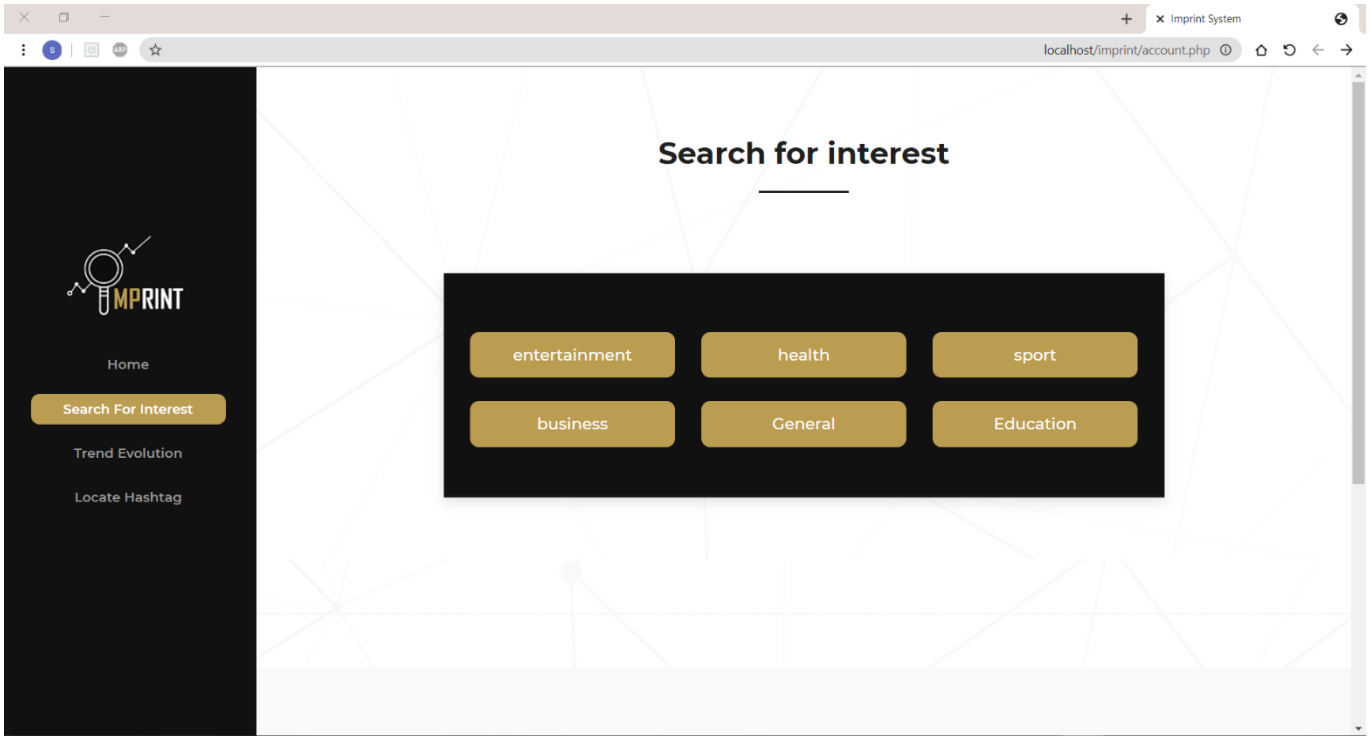
5.1.1.3 Trend Evolution

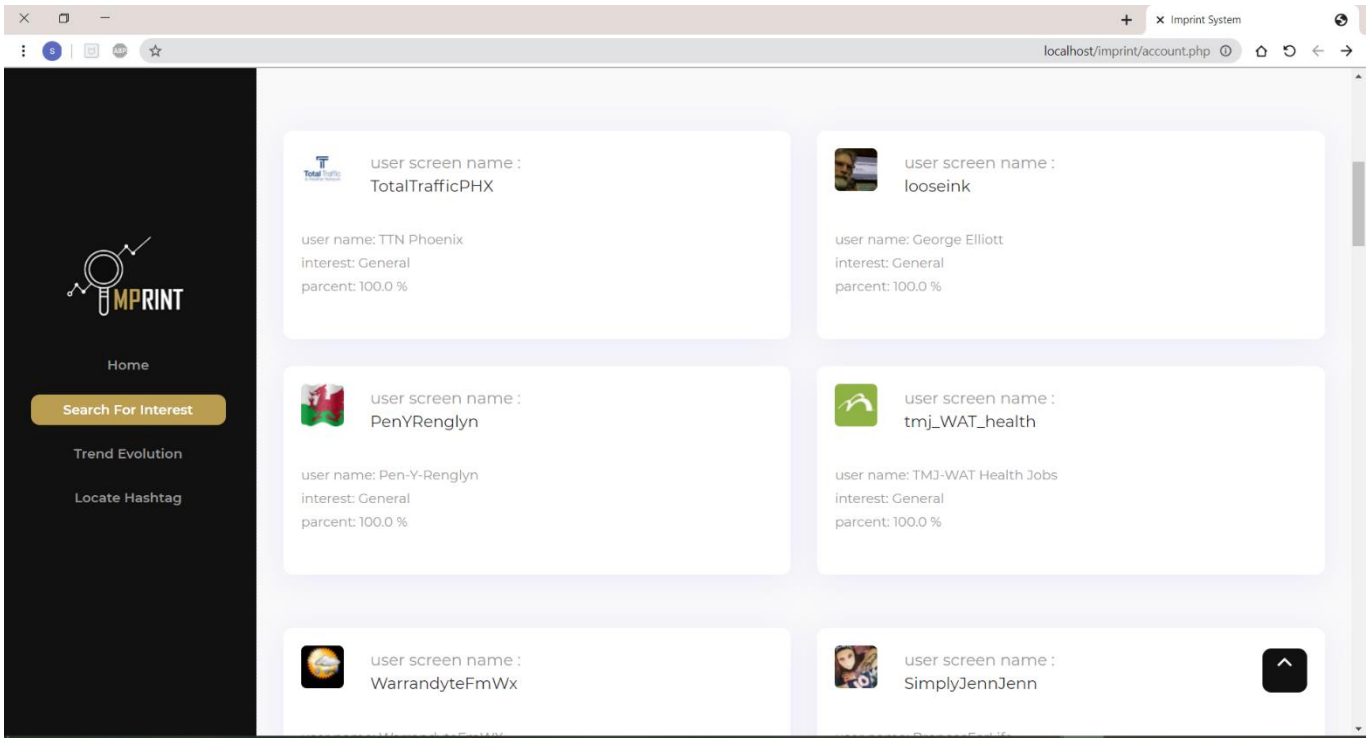
The system provides the highest trending topics, show the evolution and compute the duration it is spent in each of them in the trend list by bar Chart, X axis represent list of country order by time, y axis represents number of tweets.

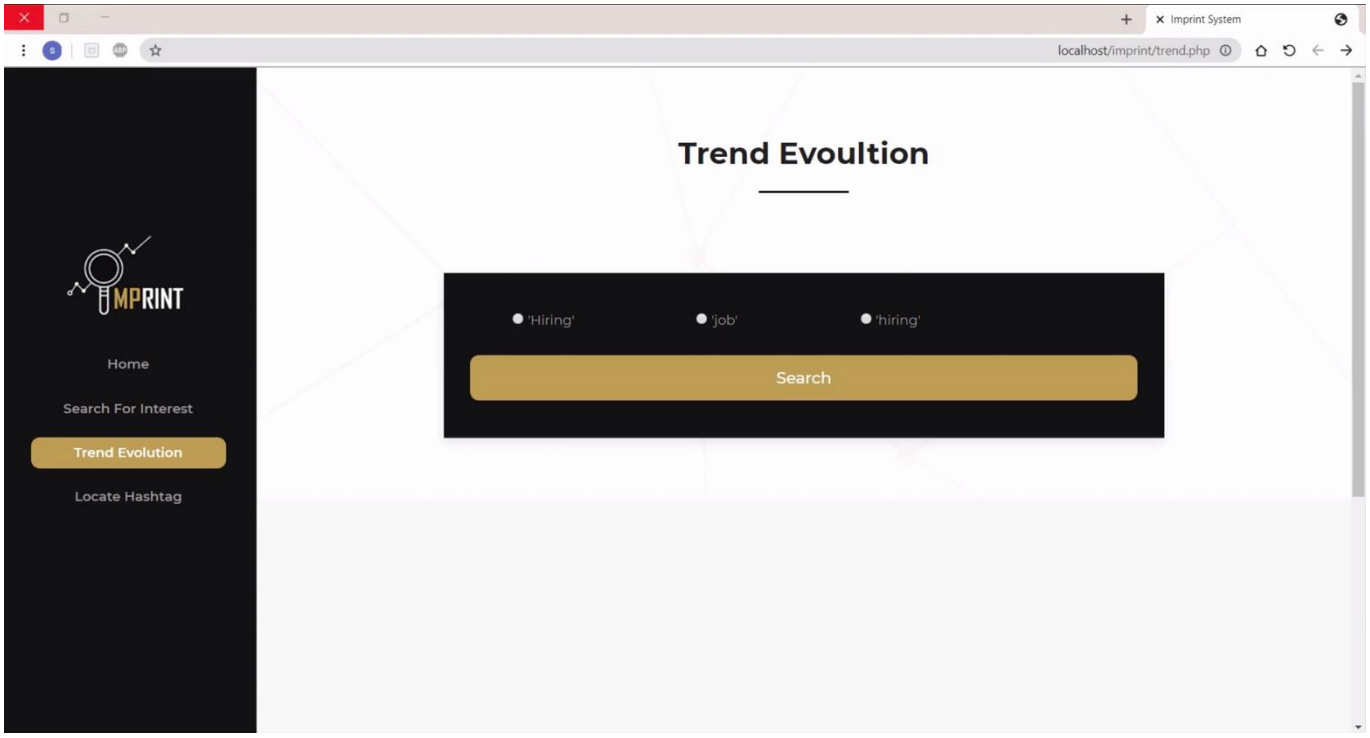
5.1.1.4 Locate Hashtag

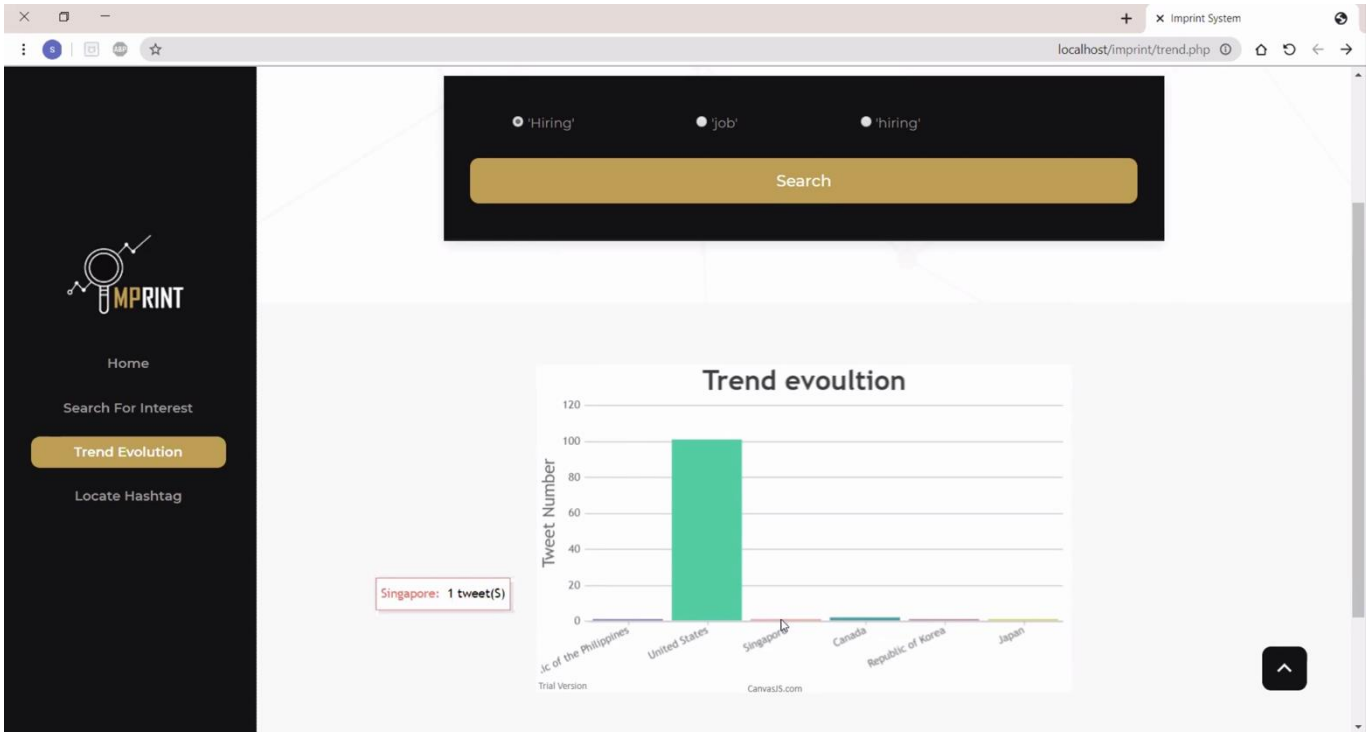
The system allow user can ask the system for specific hashtag, it extract origin which means the oldest tweet in the hashtag and specify its location, the result display in map interface. Using Openlayer: that makes it easy to put a dynamic map in any web page. It can display map tiles, vector data and markers loaded from any source. OpenLayers has been developed to further the use of geographic information of all kinds.to creates a map will import TileLayer from 'ol/layer/Tile' library and to display OSM data will import OSM from 'ol/source/OSM' library.

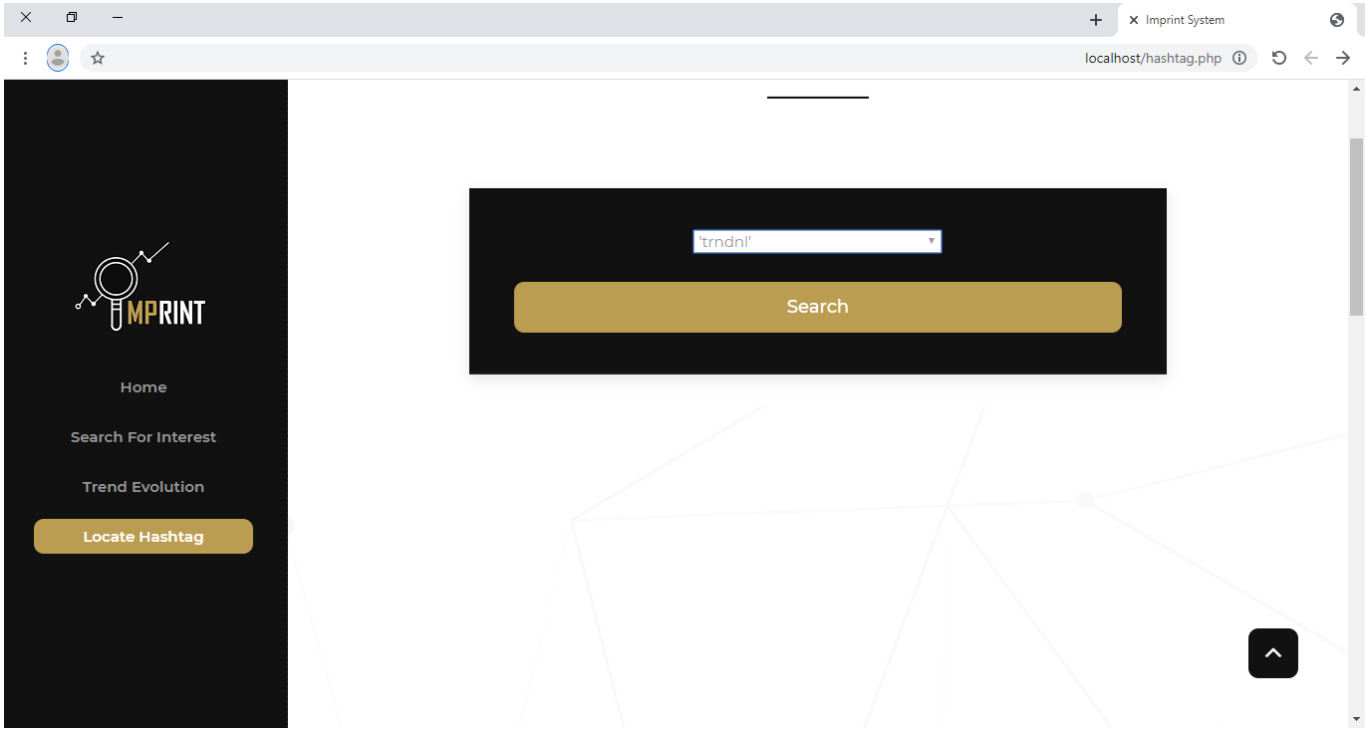
5.1.3 Web Interface:

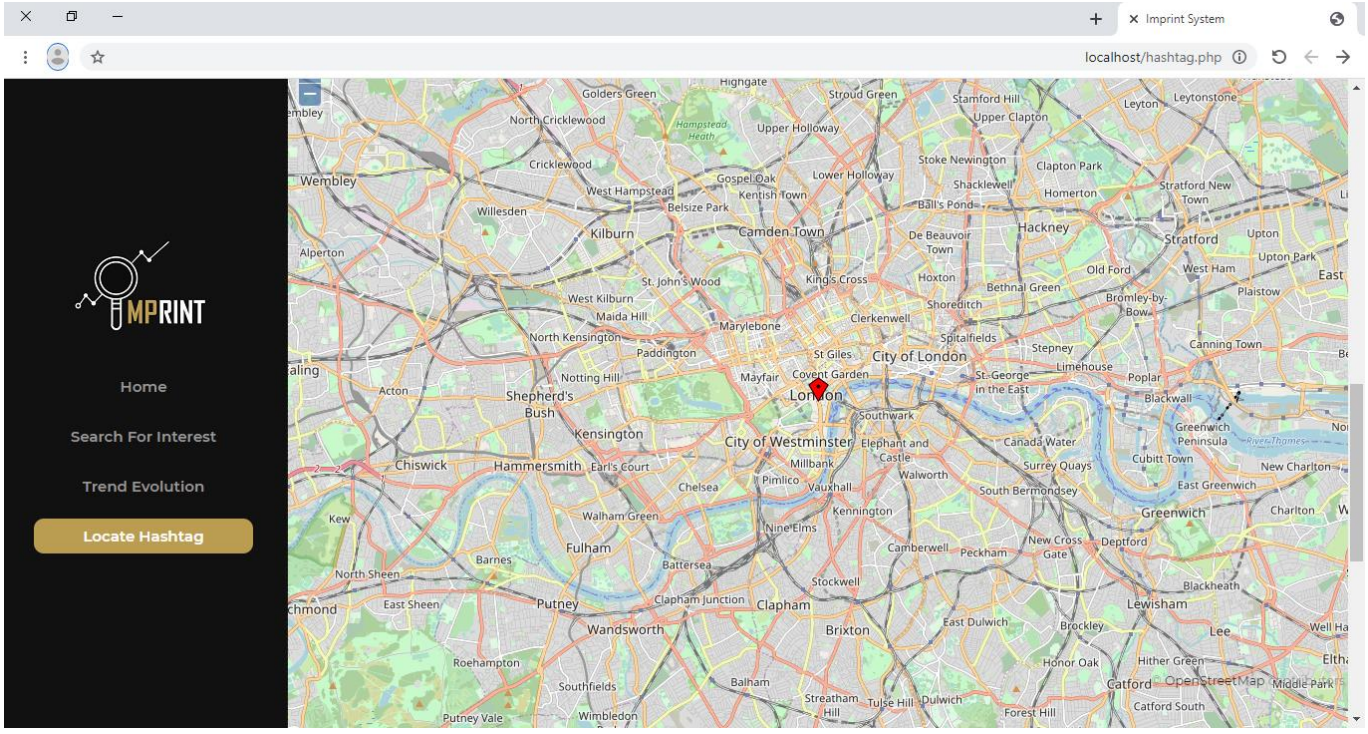





Imprint System
localhost/imprint/trend.php
MPRINT
Home
Search For Interest
Trend Evolution
Locate Hashtag
Trend Evoultion
'Hiring'
'job'
'hiring'
Search


Imprint System
localhost/imprint/trend.php
MPRINT
Home
Search For Interest
Trend Evolution
Locate Hashtag
'Hiring'
'job'
'hiring'
Search
Trend evoultion
Tweet Number
120
100
80
60
40
20
0
United States
Canada
Republic of Korea
Japan
Singapore: 1 tweet(S)
Trial Version
CanvasJS.com


Imprint System
localhost/hashtag.php
MPRINT
Home
Search For Interest
Trend Evolution
Locate Hashtag
'trndnl'
Search


Imprint System
localhost/hashtag.php
MPRINT
Home
Search For Interest
Trend Evolution
Locate Hashtag
Wembley
Kilburn
Camden Town
City of London
City of Westminster
Kensington
Chelsea
Fulham
Hammersmith
Shepherd's Bush
Notting Hill
Paddington
Marylebone
Hackney
Stratford
Greenwich
Lewisham
Brixton
Clapham
Wandsworth
Putney
Richmond
Kew
Barnes
Chiswick
Acton
Ealing
Willesden
Golders Green
Hampstead Heath
Highgate
Upper Holloway
Cricklewood
North Cricklewood
West Hampstead
Gospel Oak
Kentish Town
Belsize Park
Alperton
St. John's Wood
West Kilburn
Maida Hill
North Kensington
Mayfair
Soho
Covent Garden
St Giles
King's Cross
Hoxton
Shoreditch
Clerkenwell
Spitalfields
Bethnal Green
Stepney
Limehouse
Poplar
Canary Wharf
Canada Water
Surrey Quays
Rotherhithe
Deptford
New Cross
Peckham
Camberwell
Walworth
Elephant and Castle
Southwark
Vauxhall
Pimlico
Millbank
Westminster
Earl's Court
Walham Green
Nine Elms
Battersea
Clapham Junction
Stockwell
Brixton
East Dulwich
Forest Hill
Catford
Lee
Blackheath
Greenwich Peninsula
Cubitt Town
East Greenwich
Charlton
Woolwich
Blackwall
Canning Town
Upton Park
West Ham
Plaistow
Bromley-by-Bow
Old Ford
Stratford New Town
Leytonstone
Leyton
Stamford Hill
Upper Clapton
Clapton Park
Homerton
Hackney Wick
Shacklewell
Stoke Newington
Ball's Pond
De Beauvoir Town
Roehampton
Putney Vale
Wimbledon
Southfields
Balham
Streatham
Tulse Hill
Dulwich
Honor Oak
Hither Green
Middle Park
Catford South
OpenStreetMap

5.2 Validation:

5.2.1 validate tweet language is English:

```
def read_json(file_name):
    ''' this method for load data from json file to pandas dataframe
    1- select all tweet written in English
    2- select important information which iclude when tweet created, its id, its content(text),
        its author(user),the entiteis includ in it like(hastag name, media, other),
        its place which represent name and other important information
        that help imprint system. finally longitutde and latitude of that tweet.
    3- send the resulted dataframe to columns_extractor method.
    '''
    df_original = pd.read_json(file_name, lines=True)
    df_en_tweets = df_original[ df_original.lang == "en"]
    df_en_tweets = df_en_tweets[["created_at", "id", "id_str", "text", 'user', 'entities',
                                 'place', 'coordinates']]
    columns_extractor(df_en_tweets)
```

```

def detect_tweet_text_lang(df_cleaned_text):
    ''' this method care of the tweet has been selected and cleaned but it is not contain an english text
        using "langdetect "plugins.
        1- add new colun to assign detected language.
        2- using "for loop" to care of all tweets.
        3- using handle to make a language filter "try and except" a null, numbers,
         url links text will raise an error and make detect raise an error.
        4- in except the cleaned data frame will drop these tweets.
    '''
    df_cleaned_text['lang'] = ''
    # for loop to go through rows to detect language for text
    for i in range(len(df_cleaned_text)):
        try:
            d = detect(df_cleaned_text.at[i, 'text'])
            df_cleaned_text._set_value([i], ['lang'], d)
        except LangDetectException:
            df_cleaned_text.drop(df_cleaned_text.index[i])
    return df_cleaned_text

```

5.2.2 validate tweet text is not null:

```
def clean_tweet_text_col(df_converted):
    ''' this method clean text column using a "preprocessor as p" tweepy preprocessing
        then send the result to "clean tweet" method
    '''
    # for loop to go through rows to detect language for text len(df1) stand for number of rows on the data frame 2
    for i in range(len(df_converted)):
        # clean text and description fields
        filtered_text = clean_tweets(p.clean(df_converted.at[i, 'text']))
        # assign the cleaned new value to the data frame 2
        df_converted._set_value([i], ['text'], filtered_text)
    return df_converted
```

```python
def check_null(text):
    if str(text) == '' or str(text) == 'nan' or not str(text) or len(str(text)) == 0:
        return True
    else:
        return False


def clean_tweets(tweet):
    ''' it takes 1 tweet text:
        1- check null and assign it to nan value.
        2- after tweepy preprocessing the colon left remain after removing mentions or RT sign in the
         beginning of the tweet replace consecutive non-ASCII characters with a space, links removed, lower case
         is applied define WordPunctTokenizer() instances then append the result.
    '''
    if check_null(tweet):
        clean_tweet = 'nan'
    else:
        tweet = re.sub(r':', '', tweet)
        tweet = re.sub(r',Äª', '', tweet)
        tweet = re.sub(r'[^\x00-\x7F]+', ' ', tweet)
        user_removed = re.sub(r'@[A-Za-z0-9]+', '', tweet)
        link_removed = re.sub('https?://[A-Za-z0-9./]+', '', user_removed)
        number_removed = re.sub('[^a-zA-Z]', ' ', link_removed)
        lower_case_tweet = number_removed.lower()
        tok = WordPunctTokenizer()
        words = tok.tokenize(lower_case_tweet)
        clean_tweet = (' '.join(words)).strip()
    return clean_tweet
```

```python
    # 1- clean tweet text
    df_cleaned_text = clean_tweet_text_col(df_converted)
    # 2- drop null tweet text
    df_cleaned_text = df_cleaned_text[~df_cleaned_text.text.isnull()]
```

5.2.3 validate coordinates is not null:

```python
    # 5- drop where geo_enabled is false
    df_filtered = df_en_text[df_en_text.user__geo_enabled == True]
    df_filtered = df_filtered[~df_en_text.coordinates__coordinates__0.isnull()]
```

5.2.4 validate interest percent:

```python
        df_user = df_user.groupby(['dominant_topic_theme']).dominant_topic_theme.agg(
            count_col=pd.NamedAgg(column="dominant_topic_theme", aggfunc="count"))
        df_user.to_csv('count_interest.csv')
        df_user = pd.read_csv('count_interest.csv')
        df_user['persent'] = (df_user.count_col / user_tweet_num) * 100
        interest = df_user.loc[df_user['persent'].idxmax()].dominant_topic_theme
    user = User(user__id, user__id_str, user__name, user__screen_name, user__location,
                user__geo_enabled, user__statuses_count, user__profile_image_url, user__profile_image_url_https,
                user_tweet_num, entertainment, health, business, sport, education, general, interest, df_user['persent'])
    user.save_result()
```

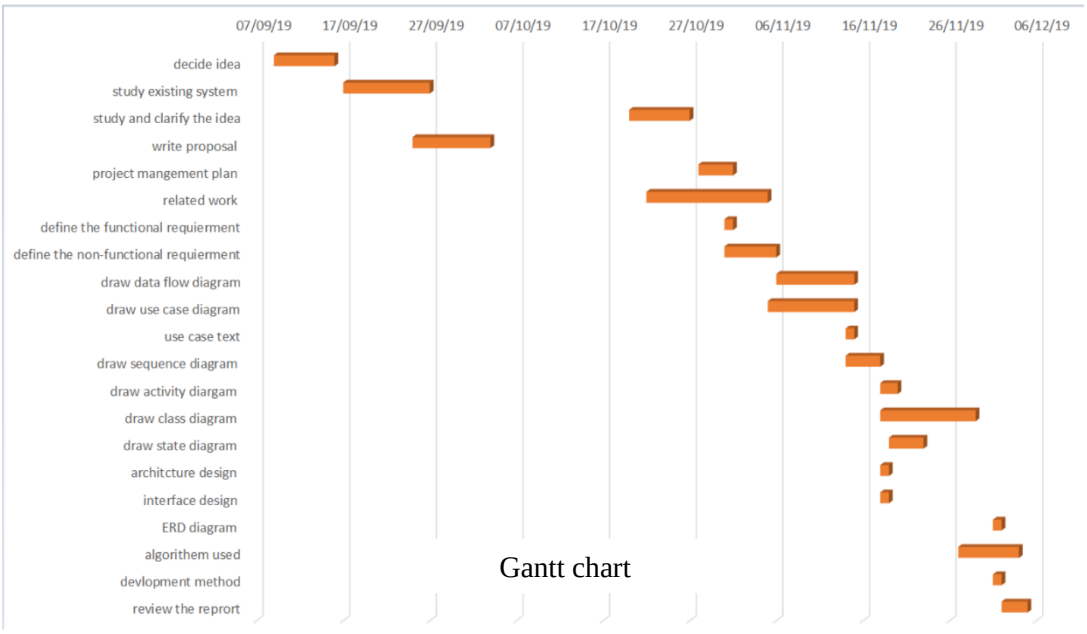
Chapter 6 conclusion:

mining information and discover knowledge from microblogs datasets is hard work but the result is worth In our designed system we analyzes the twitter dataset. The user will interact with the web-interface that can house some data mining and analysis algorithms and display results in an understandable way which is how information start, show its evolution. The web-interface will allow users to submit the following requests: Locate analyzed twitter hashtag, give a list of accounts that fit the search query, Shows the evolution of trending topics.

Appendix

A- Project Management Plan

1. Semester 1:

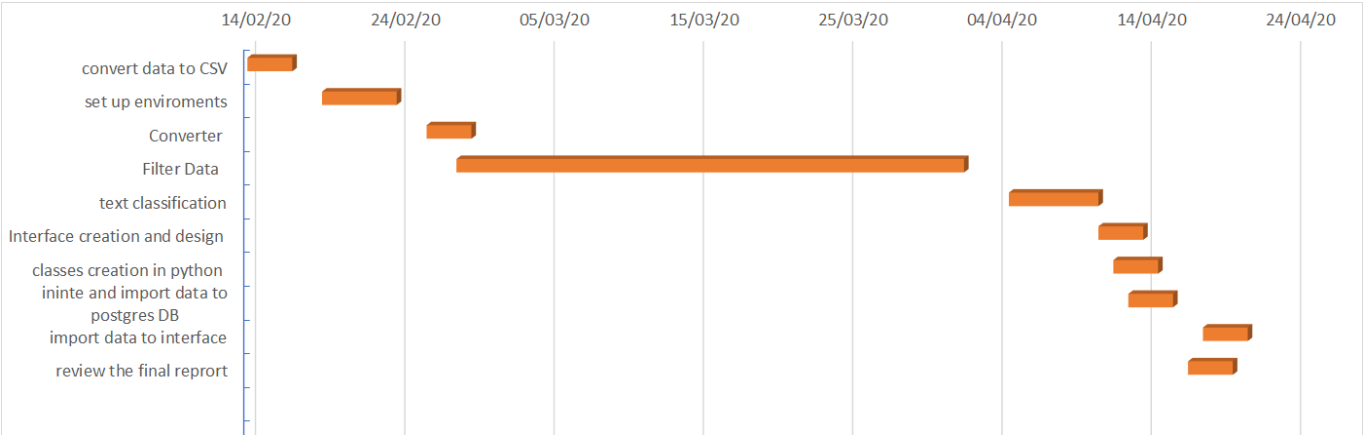


Gantt chart

primary column	Task name	duration	start date	finish date
Initiate	decide idea	7	09/09/19	16/09/19
	study existing system	10	17/09/19	28/09/19
plan	study and clarify the idea	7	20/10/19	27/10/19
	write proposal	9	25/09/19	05/10/19
	project management plan	4	28/10/19	31/10/19
	related work	14	22/10/19	05/11/19
analysis	define the functional requirement	1	31/10/19	31/10/19
	define the non-functional requirement	6	31/10/19	05/11/19
	draw data flow diagram	9	06/11/19	14/11/19
	draw use case diagram	10	05/11/19	14/11/19
	use case text	1	14/11/19	14/11/19
	draw sequence diagram	4	14/11/19	18/11/19
	draw activity diagram	2	18/11/19	19/11/19
	draw class diagram	11	18/11/19	27/11/19
	draw state diagram	4	19/11/19	22/11/19
design	architecture design	1	18/11/19	18/11/19
	interface design	1	18/11/19	18/11/19
	ERD diagram	1	01/12/19	01/12/19

algorithm	algorithm used	7	27/11/19	02/12/19
	development method	1	01/12/19	01/12/19
review	review the report	3	02/12/19	04/12/19

2. Semester 2:



primary column	Task name	duration	start date	finish date
implementation	convert data to CSV	6	11/02/20	18/02/20
	set up environments	5	19/02/20	26/02/20
	Converter	3	26/02/20	07/03/20
	Filter Data	34	28/02/20	05/04/20
	text classification	6	05/04/20	11/04/20
	Interface creation and design	3	11/04/20	13/04/20
	classes creation in python	3	12/04/20	15/04/20
	initiate and import data to postgres DB	3	13/04/20	18/04/20
	import data to interface	3	18/04/20	19/04/20
review	review the final report	3	17/04/20	19/04/20

B- References

[1] A. Magdy et al., "Demonstration of Taghreed: A system for querying, analyzing, and visualizing geotagged microblogs," 2015 IEEE 31st International Conference on Data Engineering, p. 1416, 01// 2015.

[2] Alnajran, N., Crockett, K., McLean, D. and Latham, A. (2017). Cluster Analysis of Twitter Data: A Review of Algorithms. *Proceedings of the 9th International Conference on Agents and Artificial Intelligence*.

[3] A. Zubiaga et al., "Classifying trending topics: a typology of conversation triggers on Twitter," presented at the Proceedings of the 20th ACM international conference on Information and knowledge management, Glasgow, Scotland, UK, 2011.

[4] H. Karanikas, C. Tjortjis, and B. Theodoulidis, "An Approach to Text Mining using Information Extraction," 11/09 2000

[5] N. Alnajran, K. Crockett, D. McLean and A. Latham, "Cluster Analysis of Twitter Data: A Review of Algorithms", *Proceedings of the 9th International Conference on Agents and Artificial Intelligence*, 2017. Available: 10.5220/0006202802390249 [Accessed 5 November 2019].

[6] M. Ali, R. Alsaedi, A. Alqasim, A. Umar and E. Hussin, "Product Sentiment Analysis", 2017.

[7] w. alsaedi, s. alharbi, a. alharbi, s. alsulami and s. alsaedi, "online arabic content analysis dashboard", 2018. [Accessed 8 November 2019].

[8] Sankaranarayanan, Jagan, et al. "Twitterstand: news in tweets." Proceedings of the 17th acm.

[9]"Developer", *Developer.twitter.com*, 2019. [Online]. Available: https://developer.twitter.com/en.html. [Accessed: 26- Nov- 2019].

[10] M. Russell and M. Klassen, *Mining the Social Web*. Sebastopol: O'Reilly Media, Incorporated, 2018.

[11] G. Hemantha, S. Talebi and M. K., "Users' Topic Detection from Tweets based on Keyword Extraction", International Journal of Computer Applications, vol. 167, no. 9, pp. 32-35, 2017. Available: 10.5120/ijca2017914382

[12] "susanli2016/NLP-with-Python", GitHub, 2018. [Online]. Available: https://github.com/susanli2016/NLP-with-Python. [Accessed: 21- Apr- 2020].

[13]PyPI, 2013. [Online]. Available: https://pypi.org/. [Accessed: 21- Jan- 2020].

[14] "PostgreSQL Tutorial - Learn PostgreSQL from Scratch", Postgresqltutorial.com, 2017. [Online]. Available: https://www.postgresqltutorial.com/. [Accessed: 21- Jan- 2020].

[15] "Free CSS | 3030 Free Website Templates, CSS Templates and Open Source Templates", Free-css.com, 2007. [Online]. Available: https://www.free-css.com/. [Accessed: 14- Jan- 2020].

[16] "Stack Overflow - Where Developers Learn, Share, & Build Careers", Stack Overflow, 2008. [Online]. Available: https://stackoverflow.com/. [Accessed: 08- Jan- 2020].

[17] "pandas documentation — pandas 1.0.3 documentation", Pandas.pydata.org, 2008. [Online]. Available: https://pandas.pydata.org/pandas-docs/stable/index.html. [Accessed: 05- Jan- 2020].

[18] "OpenLayers Examples", Openlayers.org, 2020. [Online]. Available: https://openlayers.org/en/latest/examples/. [Accessed: 21- Feb- 2020].

C- Work Plan

- Semester 1:

Date	Duration	Tasks	Participants
25/9	6h	Read about the idea, discussion, write in proposal.	All
1/10	3h	Write and modify in proposal.	All
2/10	1h	Write and modify in proposal.	All
4/10	2h	Modify in proposal, review.	All
5/10	0:30h	Deliver proposal.	Layan alahmadi Shahd alsayed
8/10	4h	Determine tasks, time it takes.	All
9/10	1h	chose task to do it.	All
11/10	2h	Re-search for new idea	All
28/10	3:30h	Write Project management. Search for background. discuss function.	All
31/10	2h	Write function user and system requirement.	All
1/11	2h	Related work, modify in PMP	All
5/11	6h	discus about the use case and draw it. Discus about nonfunctional requirement	All
6/11	5h	DFD drawing.	All
11/11	2h	Modify DFD.	All
14/11	6h	Modifying use case, and DFD diagrams. drawing sequence diagram.	All
18/11	4h	Modifying sequence diagram and activity diagrams. Discus and start drawing class diagram, architecture design.	All
19/11	4h	Activity diagram. State diagram.	All
20/11	5h	Review and modify diagrams	All
21/11	4h	Start prototype data base. Review and Modify on document.	All
22/11	2h	State diagram modify. Discuss prototype	All
25/9	6h	Read about the idea, discussion, write in proposal.	All
23/11	1h	Drawing, modifying class diagram.	All
24/11	3h	Review document, write prototype html codes.	All
26/11	3h	Review chapter 2,3	All

27/11	6h	Start writing in chapter 4, modifying in class diagrams	All
28/11	3h	writing in chapter 4	All
29/11	1h	writing in chapter 4	Elham Al-asmeri Shahd Al-sayed
30/11	0:30h	writing in chapter 4	All
1/12	2h	Review chapter 4, method development, Questionnaire.	All
2/12	2:40h	Review and modify chapter 4, modifying diagrams.	All
3/12	7h	Review document, write work plan.	All
4/12	5h	Review document.	All

- Semester 2:

Date	Duration	Tasks	Participants
11/2	4h	Partition data in Excel	All
15/2	6h	Save data in CSV	All
16/2	5h	Create web and search info about twitter	All
17/2	3h	Working in web	All
18/2	5h	Collecting data and convert	All
19/2	4h	Install library pandas and solve problem	All
23/2	5h	Install database postgresql	All
24/2	6h	Working in code about convert	All
25/2	6h	Working in code about convert	All
26/2	7h	Working in code about convert	All
29/2	7h	Write code convert in python	All
30/2	6h	Write code filter data	All
1/3	8h	Write code filter data	All
2/3	7h	Write code filter data	All
3/3	5h	Search how Cal acute tweet	All
4/3	8h	Install library NlTK	All
5/3	8h	Complete code filter data	All
6/3	7h	Complete code filter data	All
7/3	7h	Converter	All
15/3	5h	Poster	All
17/3	6h	Write in DB	All
22/3	7h	Modification in DB	All
30/3	5h	Modification draw DB	All
4/4	5h	Write classes	All
6/4	8h	Classification	All
10/4	8h	Classification and website	All
11/4	8h	Classification	All
12/4	7h	Write code hashtag	All
13/4	8h	How connect between DB and php and write code tweet and hashtag	All

15/3	7h	Code user and connect DB and php	All
16/4	6h	Connection	All
17/4	7h	Classes and connect	All
18/4	8h	Connect from php to DB	All
19/4	5h	Modification documents	All