Final Project in Software Engineering Application Design Document

An information flow control model for online social network

Omer Sella
Sagiv Mapgavker
Alexander Chinyan

2018

Table of Contents

Chapter 1

Use-cases

The main use-cases for our software system:

1. Log-in to the system with Facebook credentials:

Actors:

- User
- System

Precondition:

- The system is running on stable server
- The user connected to Internet, using HTML support browser and navigated to system's website via correct URL
- The user has a Facebook account

Postcondition:

- The user is connected to the system, delegated authority to the system to crawl his public data in Facebook

Main scenario:

1) The user press the "Crawl Facebook" button
2) Insert Facebook credentials.
3) The user is already logged in to Facebook
4) The user agrees to connected to the system via his Facebook account
5) The user connects to the system - Postcondition achieved

Alternative scenario:

1) The user press the "Crawl Facebook" button.
2) The user is **not** logged in Facebook.
3) The system will show a login window.
4) The user will enter his Facebook credentials.
5) The user connects to the system - Postcondition achieved.

More alternative scenario:

1) User entered invalid Facebook credentials while login to Facebook - the system will present a suitable message and allow the user to re-enter the details.

2. Log-in to the system with Twitter credentials:

Actors:
- User
- System

Precondition:
- The system is running on stable server
- The user connected to Internet, using HTML support browser and navigated to system's website via correct URL
- The user has a Twitter account

Postcondition:
- The user is connected to the system, delegated authority to the system to crawl his public data in Twitter

Main scenario:
1) The user press the "Crawl Twitter" button
2) Insert Twitter credentials.
3) The user is already logged in to Twitter
4) The user agrees to connected to the system via his Twitter account
5) The user connects to the system - Postcondition achieved

Alternative scenario:
1) The user press the "Crawl Twitter" button.
2) The user is **not** logged in Twitter.
3) The system will show a login window.
4) The user will enter his Twitter username.
5) The user connects to the system - Postcondition achieved

More alternative scenario:
1) User entered invalid Twitter credentials while login to Twitter - the system will present a suitable message and allow the user to re-enter the details.

3. Upload CSV file which contains main algorithm input:

Actors:
- User
- System

Precondition:
- The user navigated to system's website via correct URL.
- The user doesn't have to be logged in to the system.

Postcondition:
- The CSV file uploaded to the server side and ready to execute the system's main algorithm.

Main scenario:
1) The user press the "Upload CSV file" button.
2) There is window which the user can choose the file to upload from his computer.
3) The user choose from his computer which file to upload
4) The user press on "Upload" button.
5) The file transfer from the user's endpoint to the system's endpoint on the deployed server.
6) The system validated the file:
    - checks if it CSV file.
    - checks if it fit to the structure of the CSV file fit the algorithm input requirements.
    - checks if the file is malicious.
7) The system show an approval message to the user that the file is uploaded successfully.

Alternative scenario:
1) The user press the "Upload CSV file" button.
2) There is window which the user can choose the file to upload from his computer.
3) The user choose from his computer which file to upload.
4) The user press on "Upload" button.
5) The file transfer from the user's endpoint to the system's endpoint on the deployed server.
6) The system validated the file:
    - checks if it CSV file.
    - checks if it fit to the structure of the CSV file fit the algorithm input requirements.
    - checks if the file is malicious.
7) **The file uploaded is not valid**
8) The system will show a message that the file is not valid with explanation
9) The system will ask the user to re-upload the file

More alternative scenario:
1) The file upload process crashed while transferring - The system will show an error message and allow the user to re-try to upload the file
2) The file uploaded is considered as malicious - the system will disconnect the user from the system and ban his IP address.

4. <u>Running crawling on Facebook data of user to generate CSV file as input:</u>

Actors:

- User
- System
- Facebook's data

Precondition:

- The user logged in to the system via "login with Facebook" API

Postcondition:

- The system generate a CSV file according the user social network (Facebook) data:
    - Data on user's first circle friends - direct friends
    - Data on user's second circle friends - direct friends of the first circle
- The CSV file is fit to the main algorithm of the system (use case number 6)

Main scenario:

1) The user press on "Crawl with Facebook" button
2) The system start to crawl his social network data - (Use-case 4.1)
3) The system will save according all user's first circle friends these data in the CSV file:
    - Total number of friends (TF)
    - Age of user account (AUA)
    - Amount of mutual friends (MF)
    - Friendship duration (FD)
4) The system will save according user's second circle friends' public data in the CSV file:
    - Total number of friends (TF)
    - Age of user account (AUA)
    - Amount of mutual friends (MF)
    - Friendship durations (FD) with connecting friends

Alternative scenario:

1) The user press on "Crawl with Facebook" button
2) The system start to crawl his social network data - (Use-case 4.1)
3) **The crawling interrupted** in the middle of process
4) The system will ask the user if he want to re-run this use-case:
    a) If accept - the system will re-run
    b) If not - The system will continue with the already crawled data

More alternative scenario:

1) Friend from first\second circle data is not public - The system will use educated guesses to fill the missing data - if possible - to create the most satisfying CSV file to the main algorithm

4.1. Crawl user's Facebook data:

Actors:

- User
- System
- Facebook's data

Precondition:

- The user pressed on "Crawl with Facebook" button

Postcondition:

- The system have the user's Facebook's data that fit the CSV file format

Main scenario:

1) The system will login to Facebook with user's credentials
2) The system will get user's friends list
3) The system will get all user's friends:
    a) Total number of friends (TF)
    b) Age of user account (AUA)
    c) Amount of mutual friends (MF)
    d) Friendship duration (FD)
4) For each User's Friend the system will get his friend list
5) For each friend of friend, the system will get:
    a) Total number of friends (TF)
    b) Age of user account (AUA)
    c) Amount of mutual friends (MF)
    d) Friendship duration (FD) with connecting friend

4.2. Crawl user's Twitter data:

Actors:

- User
- System
- Twitter's data

Precondition:

- The user pressed on "Crawl with Twitter" button

Postcondition:

- The system have the user’s Twitter’s data that fit the CSV file format

Main scenario:

1) The system will login to Twitter with user’s credentials
2) The system will get user’s friends list
3) The system will get all user’s friends:
    a. Total number of friends (TF)
    b. Age of user account (AUA)
    c. Amount of mutual friends (MF)
    d. Friendship duration (FD)
4) For each User’s Friend the system will get his friend list
5) For each friend of friend, the system will get:
    a. Total number of friends (TF)
    b. Age of user account (AUA)
    c. Amount of mutual friends (MF)
    d. Friendship duration (FD) with connecting friend

5. <u>Choose the input parameters to the main algorithm:</u>

Actors:

- User
- System

Precondition:

- The system have the user’s CSV input file via crawling or via upload

Postcondition:

- The system have the user’s preferences about the input parameters to the main algorithm

Main scenario:

1) The system will present the user list of parameters that will be considered while running the algorithm:
    a) first circle’s friends:
        i) Total number of friends (TF)
        ii) Age of user account (AUA)
        iii) Amount of mutual friends (MF)
        iv) Friendship duration (FD)
    b) second circle’s friends:
        i) Total number of friends (TF)

ii) Age of user account (AUA)
iii) Amount of mutual friends (MF)
iv) Friendship durations (FD) with connecting friends

2) The user will choose the parameters (Check boxes) that he wants to run algorithm according to.

Alternative scenario:

1) The user didn't choose any parameter - The system default state is to run the algorithm based on all parameters

6. <u>Graph choose by the user:</u>

Actors:

- User
- System

Precondition:

- The system finished successfully running the algorithm and presenting 3 options of graphs for the user to choose

Postcondition:

- The system will present selected graph

Main scenario:

1) The user choose one of the options
2) The system will show selected graph

7. <u>Graph filter according user selection:</u>

Actors:

- User
- System

Precondition:

- The system presents a graph to user

Postcondition:

- The system show the graph's part that the user selected

Main scenario:

1) The system presents options for filtering the graph:
    a) All connections
    b) Only bad connections
2) The user selects one of the options
3) The system will present the graph's part that the user selected

Alternative scenario:

1) The user didn't select any option - The system will continue present the default graph
2) The filtered graph is empty - The system will present suitable message

8. <u>Zooming in\out the presented graph:</u>

Actors:

- User
- System

Precondition:

- The system presents a graph to user

Postcondition:

- The presented graph is zoomed in \ zoomed out as the user requested.

Main scenario:

1) The user points the mouse on the area that he want to zoom in\out
2) The user use his mouse scroll to zoom in\out
3) The system enlarge\reduce the area of user's point

Alternative scenario:

1) The user reach the limit - Graph is too small\big - The system will limit the zoom in\out action

9. <u>Showing info on each node in the graph:</u>

Actors:

- User
- System

Precondition:

- The system presents a graph to user

Postcondition:

- The system will present the information of the node (user’s friend) on window at the right down corner.

Main scenario:

1) The user stand on one of the nodes that represent user’s friend in first\second circles of friends.
2) The system will show the friend’s details that was input for the algorithm-
    a) For friends:
        i) Id
        ii) Name
        iii) Total number of friends (TF)
        iv) Age of user account (AUA)
        v) CF(connected friend)
        vi) Amount of mutual friends (MF)
        vii) Friendship duration (FD)
        viii) TSP
        ix) Level(first or second circle)
    b) Determination if the user is Adversary or Acquaintance

Alternative scenario:

1) The information of the user is based on suggested guess - The information in this node will include a notify that the information is inferred

10. <u>Disconnecting from the system:</u>

Actors:

- User
- System

Precondition:

- The user is connected to the system

Postcondition:

- The user is disconnected from the system
- There is no saved data of the user on the server side

Main scenario:

1) The user press the ‘X’ browser button.

11. <u>Exporting CSV file:</u>

Actors:
- User
- System

Precondition:
- The user is connected to the system
- The Crawler has done scanning and gathering data
- The Crawler has produces CSV file contains the data

Postcondition:
- The user has downloaded the file
- The file contains all the data that was collected
- The data is anonymous and encrypted(if clicked on anonymous checkbox)

Main scenario:
1) The user press the “Download CSV file” button
2) The file transferring to user’s computer desktop or sent to mail.

12. <u>Adding manually data to the system:</u>

Actors:
- User
- System

Precondition:
- The user is connected to the system
- The user at the main page of the system

Postcondition:
- The data has successfully added to the system and presented in the graph.

Main scenario:
1) The user press the “Add Data Manually” button
2) Window is present to add all relevant fields
3) The user press the “Add Node” button
4) The data is successfully added to the system

Alternative scenario:
1) The system failed to add the data to the system
2) Exception has presented to the user
3) The user may try again to add information

12.1 System added the additional data to the JSON file:

Actors:

- System
- User

Precondition:

- The user has entered additional data

Postcondition:

- The additional data has been added to the CSV file

Main scenario:

1) The user has pressed 'Add Data Manually" button
2) The user entered all data
3) The user has pressed the 'Add Node' button
4) The system add data in accordance 12.1.1/12.1.2

12.1.1 Data added to the uploaded JSON file:

Actors:

- System

Precondition:

- The system adds data to JSON file that was uploaded by the user

Postcondition:

- The data has been added to the JSON

Main scenario:

1) The data to be added is inserted to the system
2) The user press on "Download Data"
3) The system is adding the data to the JSON file
4) The data has been added

Alternative scenario:

1) The data to be added is inserted to the system
2) The system is adding the data to the JSON file
3) The data has **not** been added
4) The new JSON file does **not** contain the new data

12.1.2 Data added to the produced JSON file:

Actors:

- System

Precondition:

- The system adds data to JSON file that was produced by the system

Postcondition:

- The data has been added to the JSON

Main scenario:

1) The data to be added is inserted to the system
2) The user press on “Download Data”
3) The system is adding the data to the JSON file
4) The data has been added

Alternative scenario:

1) The data to be added is inserted to the system
2) The system is adding the data to the CSV file
3) The data has **not** been added
4) The new CSV file does **not** contain the new data

12.2 Output graph contains the added data:

Actors:

- System

Precondition:

- The new data has been entered to the system

Postcondition:

- The system added the new data to the output graph

Main scenario:

1) The data to be added is inserted to the system
2) The system is adding the data to the output graph

Alternative scenario:

1) The data to be added is inserted to the system
2) The system has failed to add the new data to the output graph
3) The output graph contains only the old data

Chapter 2

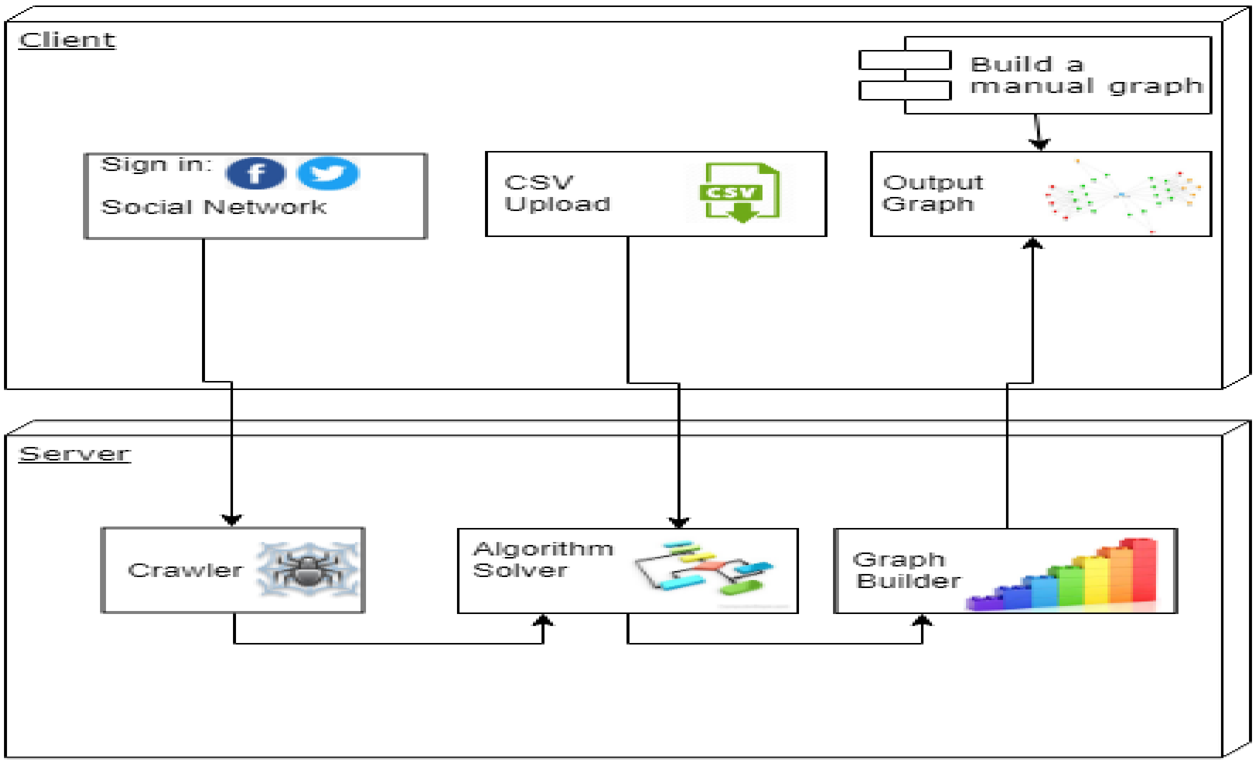
System Architecture

Our system is a web platform that provides social networks analyzes service to the user.
The system can be divided into two main layers, Presentation layer that contains an interactive GUI for comfortable use of the system and the Logical layer that responsible for the calculation and data handling.

The presentation layer is responsible for communication with the user, it is located at client side. This layer contains the 'Crawl with Facebook' option that allows the user to connect to his social network account or give his twitter account nickname in order to perform scanning.
The user can upload a CSV file to the system is also a responsibility of the presentation layer, the user can also choose to add data manually.
Also the presentation layer responsible for output graphs according to the user preferences.
The modules that responsible for visualization are located in this layer.

Another layer is the Logical layer that responsible for the calculation and data creation.
The logical layer are located on the server side.
This layer contains the Crawlers components that is the main data resource for our system.
We built a Facebook crawler and a Twitter crawler to give our users support for the most popular social media nowadays.
Another component that belongs to the logical layer is the AlgorithmSolver , this component are responsible to analyze the data stream that it receives. We also have a component that transform a csv to Json file.

The data may come from a CSV file that has been uploaded by the user or by CSV file that has been produced by the Crawler as a direct result of scanning the social network, Or JSON file that has been produced manually.
The Crawler is the component that produces .csv files.
The logical layer also contains the Graph Builder component, the graph builder gets input stream from the algorithm solver and best on the data it receives this component produces the necessary data that the output graph needs.


Client
Build a manual graph
Sign in:
Social Network
CSV Upload
Output Graph
Server
Crawler
Algorithm Solver
Graph Builder

Chapter 3

Data Model

Our system is engaging with large amount of data, we support both collecting data from popular social networks, e.g. Facebook and Twitter, and analyzing CSV files with suitable structure that contains relevant data. Another option is to make manually data by ‘Add Data Manually’ and make a JSON file.
Therefore the main objects in the system are objects that hold representation of gathered data for further purposes such as display the data to the user as an output.
No data is stored in a relational database, we use personal data hence we make sure that there isn’t any potential risk to safety.
The CSV and JSON is being anonymized(if choosen) to protect the personal data of each user that was scanned.

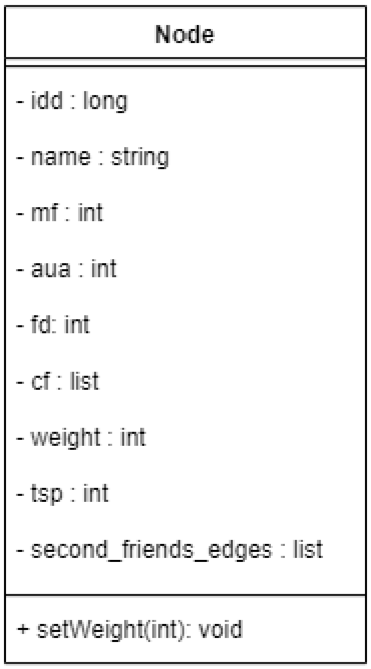
3.1 – Description of data objects

CSV file and JSON file

The CSV (or JSON) file contains the necessary data, contains information about every user that was scanned(or produced manually).
The system process the csv (or JSON) file and building a graph.
The file contains the needed information for the algorithm to calculate the score.
From the csv (or JSON) file we create a node for each user, every connection is represented by an edge; after that the system starts building the graph.
Also, after collecting data by the Crawler we produce a CSV (or JSON that produced manually) file that can be downloaded by the user for future use of our system.

The CSV (or JSON) file contains the following fields:
- User ID
- User Name
- Total number of friends (TF)
- Amount of mutual friends (MF)
- Age of user account (AUA)
- Friendship duration (FD)
- Connecting Friends (CF)

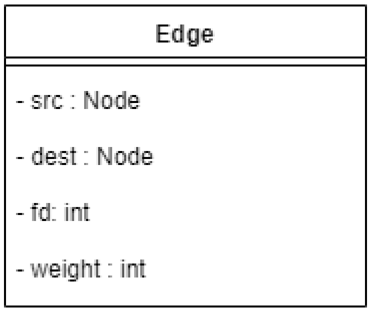
Node



The node object represent every user of the social network that has been scanned and analyzed; the node will be part of our output graph.

- Id – each node has an unique id number
- Name – each node has a name.
- mf – each node has a mf(num of mutual friends- int).
- aua- each node has an age of user account- num of month
- fd- each node has a Friendship duration- num of month
- cf- connecting friend id(list)
- weight – each node will get a score according to our algorithm
- tsp- total sharing probability, the score of node.
- Second_friends_edges- list of edges with second circle friend.
- setWeight- change the weight of node.

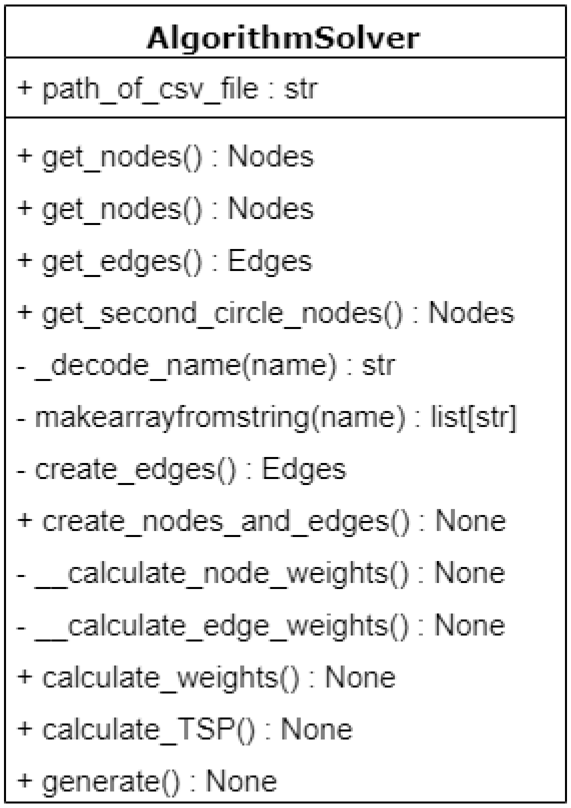
Edge



The edge object represent a connection between two users in the social network.
The edge will be shown in the output graph and connect between 2 nodes.

- src – id number of the source node.
- dest – id number of the destination node.
- fd- each node has a Friendship duration(between src and dest)- num of month
- weight – each edge has a score that has been given by the algorithm, we are using it's weight to make future decisions based on the weights of the edges and nodes.

Algorithm solver


AlgorithmSolver
+ path_of_csv_file : str
+ get_nodes() : Nodes
+ get_nodes() : Nodes
+ get_edges() : Edges
+ get_second_circle_nodes() : Nodes
- _decode_name(name) : str
- makearrayfromstring(name) : list[str]
- create_edges() : Edges
+ create_nodes_and_edges() : None
- __calculate_node_weights() : None
- __calculate_edge_weights() : None
+ calculate_weights() : None
+ calculate_TSP() : None
+ generate() : None

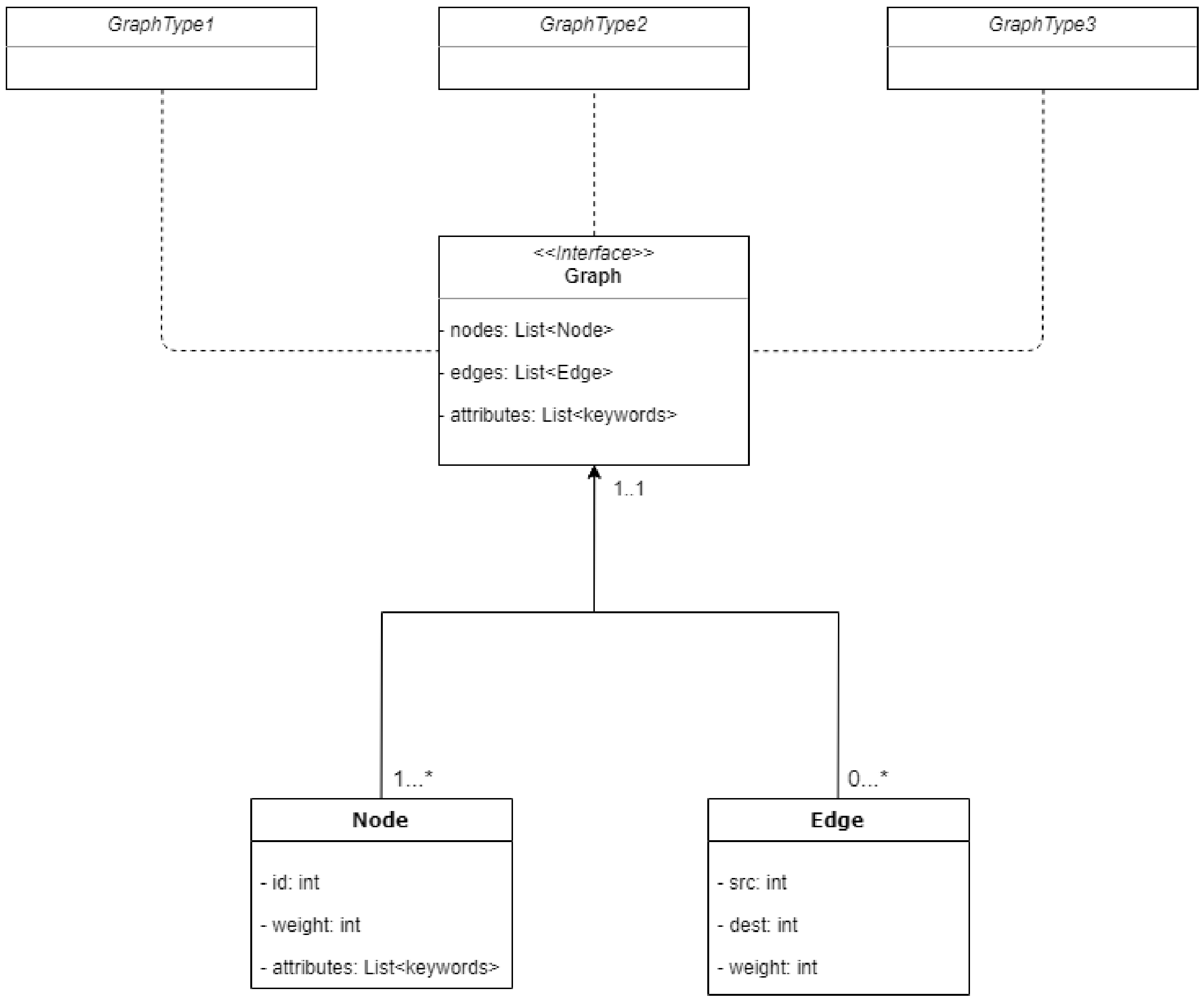
Responsibilities: receives CSV as input, process the CSV file and creates Node for each user; each connection between user is represented by an Edge.
Also, this class responsible to implement the theoretical part of our project; In this class we apply the theory described by our advisors' articles.

- Main fields:
    - nodes: List<Nodes> - list of nodes
    - edges: List<Edges> - list of edges
- Main methods:
    - void generate():
        - Description: the main method of this class, this method processes and generates all the data needed to build the output graph.
          Analyses the CSV file that has been received.
    - create_nodes_and_edges():
        - Description: this method called from generate method and creates nodes and edges from the CSV file.
          It updates the nodes and edges lists.
    - calculate_weights():
        - Description: this method calculates weight for each node and edge, the 'weight' field is updated for each object.
    - calculate_TSP():
        - Description: in this method the TSP field is calculated for each Node according to the formula described in the article.

Json

We are using Json file to save the data about nodes and edges in order to present the output graph to the user. Json files helps the different layers to communicate and transfer data between each other. The presentation layer loads the Json file and present the graph.

3.2 - Data Objects Relations



Graph is consists of nodes and edges, there is at least one node that is the ego node.
There are 3 types of graph, each type is a different representation of the graph interface.
Every type implements the interface in the way that it needs.

3.3 - Databases

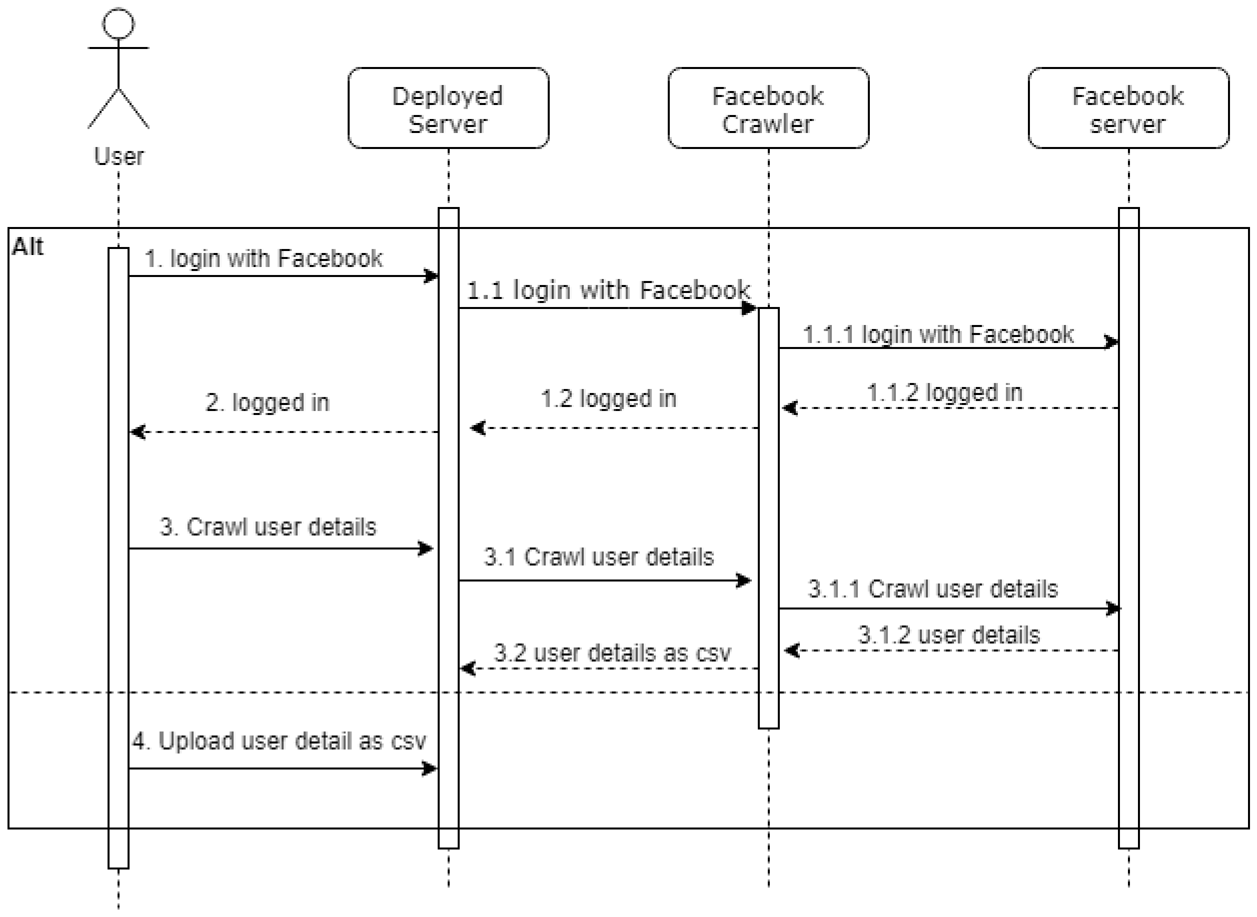
Our system don’t use any database, we don’t save any data on our servers.
We produce CSV file that contains all the necessary data, the data is anonymous and the file is encrypted.

Chapter 4

Behavioral Analysis

4.1 Sequence Diagrams

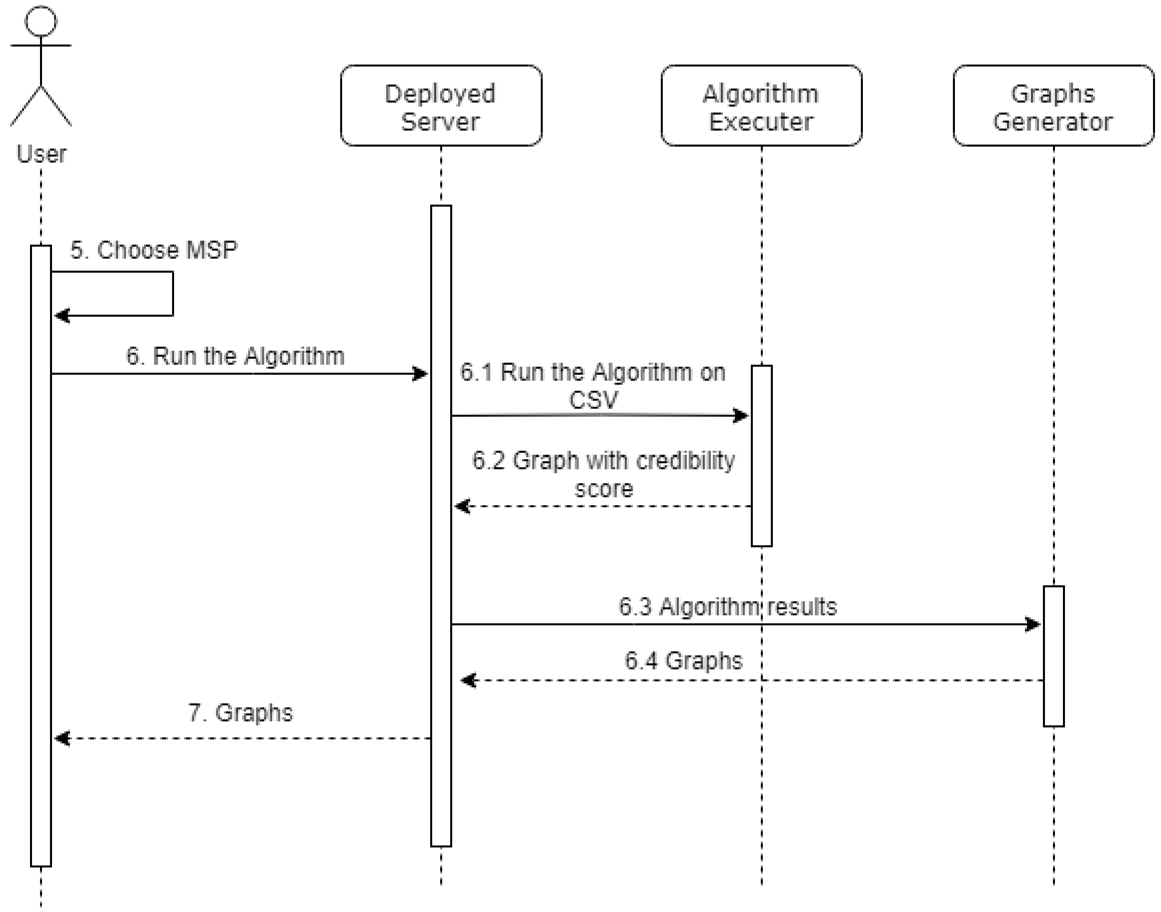
CSV Gathering sequence:



At this part there are 2 options of generating csv file input to our algorithm:

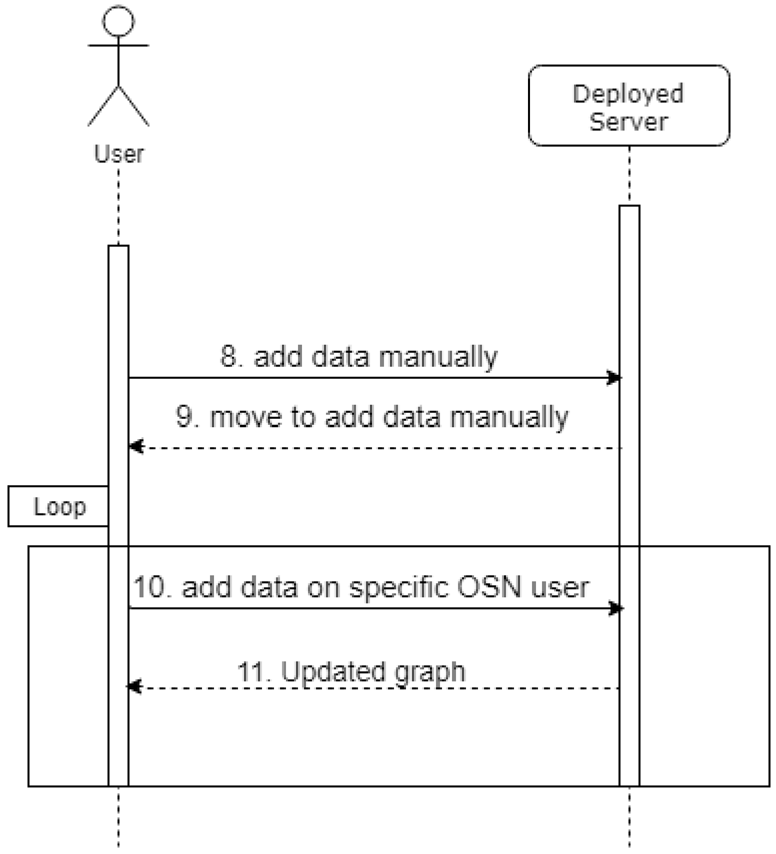
1. Generating data via Facebook crawler.
2. Uploading already generated CSV file

2) Algorithm execute sequence:



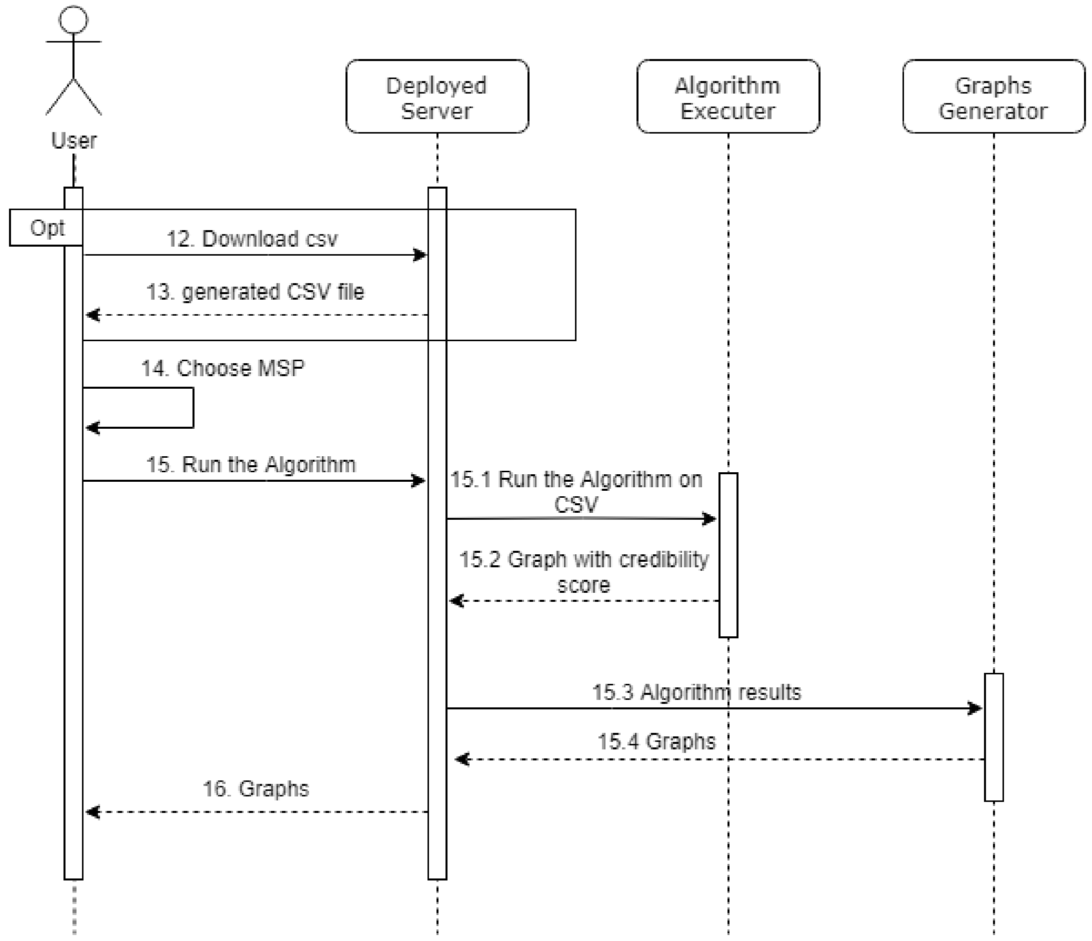
This sequence describe the flow of the system after the data gathering is over.
Running the algorithm based on the csv file and generate and represent graphs to the user interface.
There is a option to the user to download generate CSV for later processing.

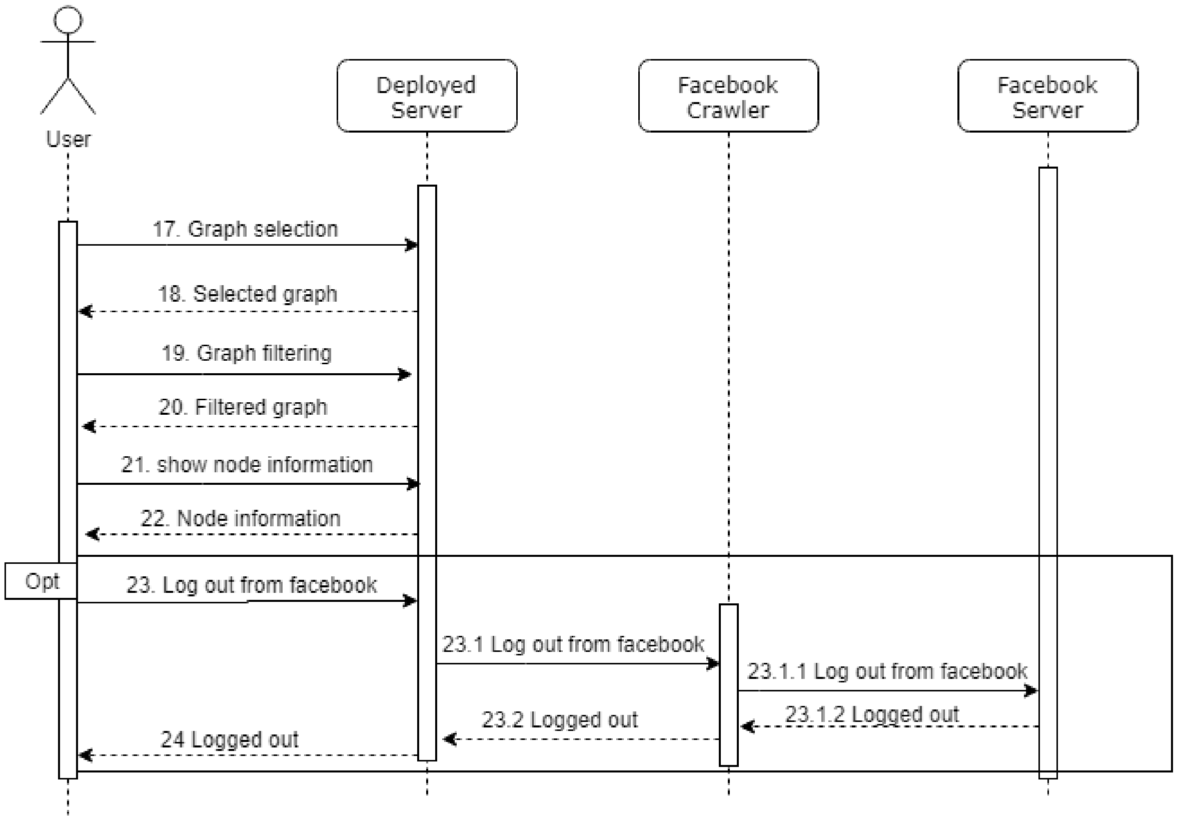
3) <u>Add graph manually:</u>


User
Deployed Server
8. add data manually
9. move to add data manually
Loop
10. add data on specific OSN user
11. Updated graph

1. The system will allow to add a graph manually.
    1.1. There is a page that allows you to insert the data on a node and then insert it into a graph.
    1.2. Each node insert re-displays the current graph.
    1.3. It is possible to save the generated graph to a .csv file for future use.

4) Working with graphs and logout sequence:





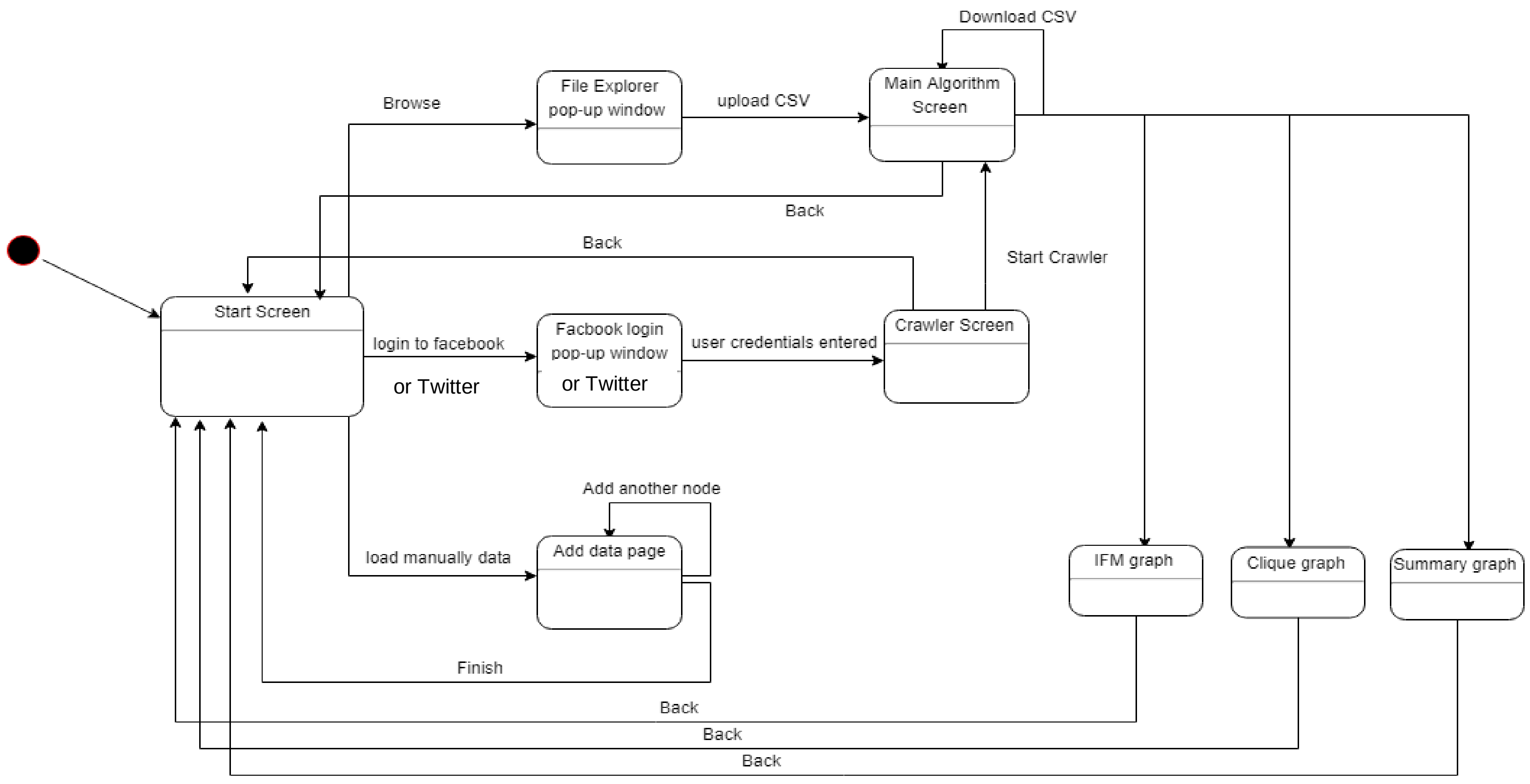
This sequence diagram represent the manipulations that the user can do on graph that the system generated according the execution of the algorithm.
If the user is logged-in to the system via his Facebook account, the user will able to log out.

4.2 - Events

Event	Description	Action
Add manually data	The user press on “Add manually data” button in “Start screen”.	The user is being transferred to “Add manually data” screen.
Sign in Facebook or Twitter and crawl	The user press on “Crawl with Facebook” button in “Main screen”.	After the user fill the username and password,the user signing in to Facebook or Twitter and is being starts crawling.
Browse CSV file	The user press on “Browse” button in “Start screen”.	File explorer opened, the user navigate to the correct path to CSV file and click open, the system loads the CSV file and the user is being transferred to “Main algorithm screen”.
Add new Node	The user fill the fields (except ID field that automatically filled) of the node and press of “Add Node” button in “Add data manually screen”.	The node added to JSON file by the system, if this is the first node, the system open new JSON file and then add the node.
Finish	The user press on “Download data” button in “Add manually data screen”.	The user gets the new JSON file, that contains the new data.
Start Crawler	The user press on “Start to crawl” in “Main Screen”.	The crawler is scanning the social network of the logged in user. The crawler is gathering all the necessary data to future calculation. In addition the system generates new CSV file (if doesn’t exist) that contains the new data. The user is being transferred to “Main screen”.
Back from “upload existed CSV file” / “add manually data” screen	The user press on “Back” from “upload existed CSV file” / “add manually data” screen.	The user is being transferred to “Main screen”.
Download CSV	The user press on “Download CSV” button in “CSV scanned” screen.	The browser downloads the CSV file from the server.
Start algorithm	The user choose the parameters to run,the graph model and then press on “Recalculate” button.	The system calculate the algorithm, build the relevant graph(according to the graph model), in the relevant screen the graph is displayed.

Back from graph	The user press on "Back" from one of the Graph screens.	The user is being transferred to "Main screen".

4.3 - States

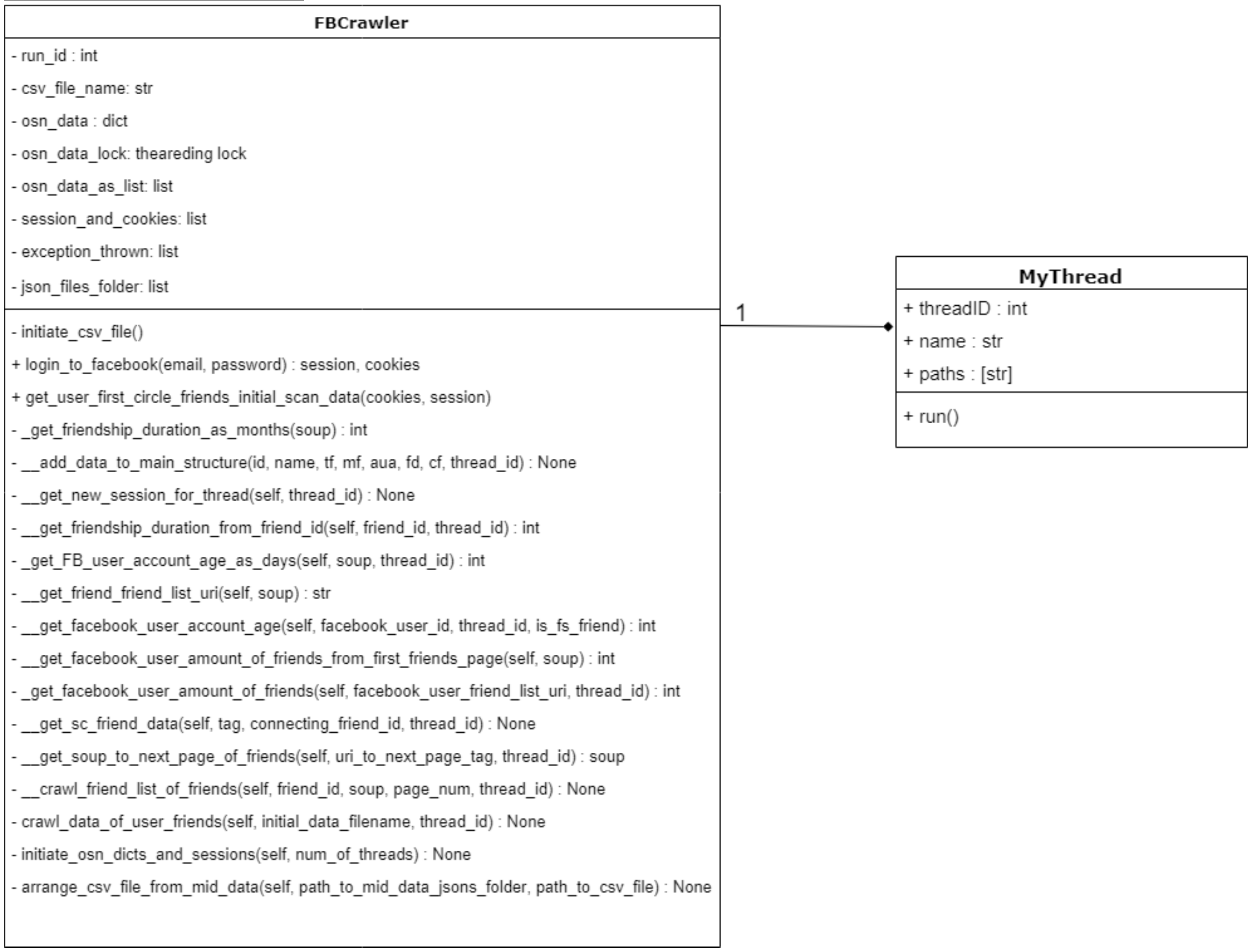


Chapter 5

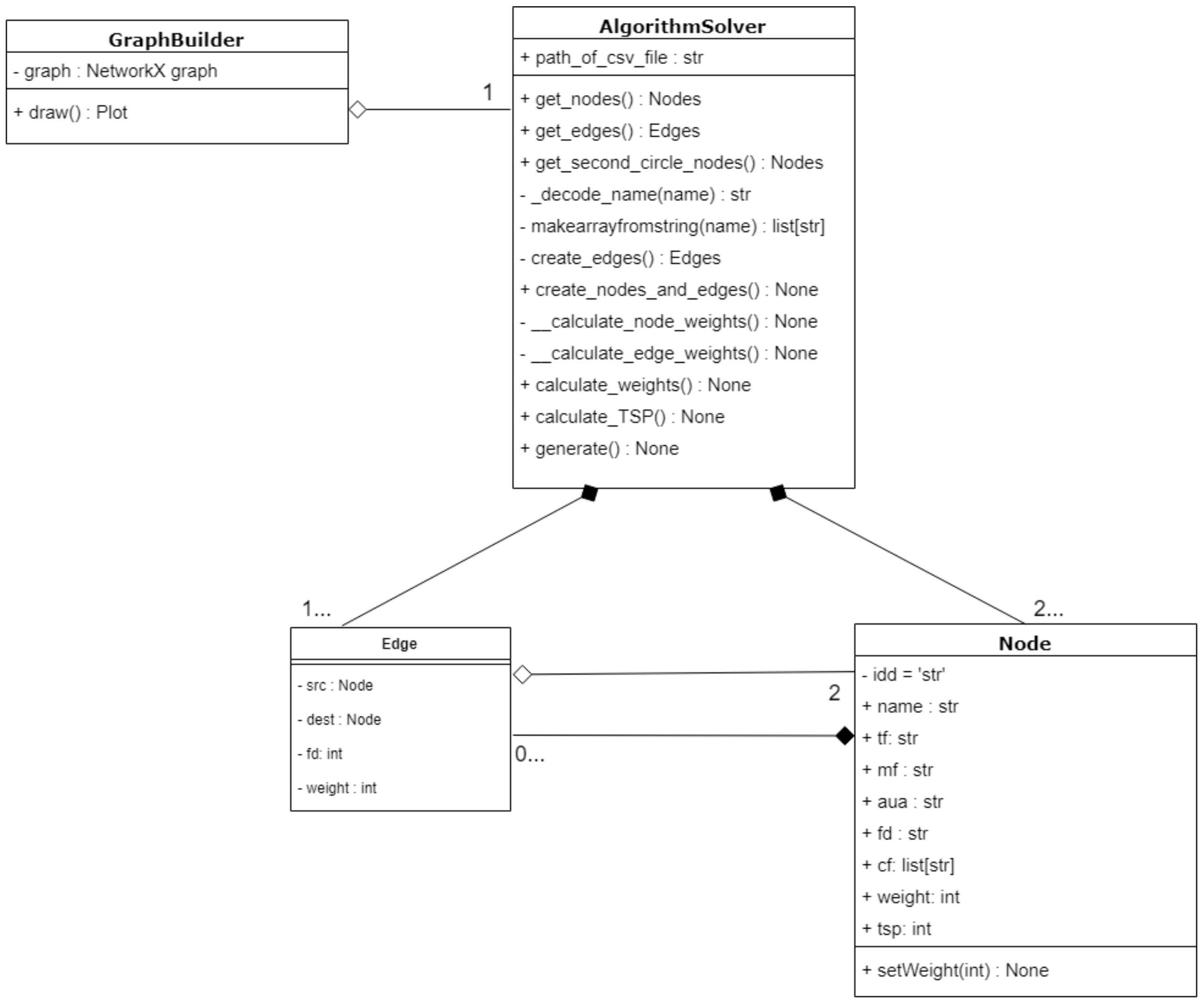
Object Oriented Analysis

5.1 - Class Diagrams

Crawler class diagram:


FBCrawler
- run_id : int
- csv_file_name: str
- osn_data : dict
- osn_data_lock: theareding lock
- osn_data_as_list: list
- session_and_cookies: list
- exception_thrown: list
- json_files_folder: list
- initiate_csv_file()
+ login_to_facebook(email, password) : session, cookies
+ get_user_first_circle_friends_initial_scan_data(cookies, session)
- _get_friendship_duration_as_months(soup) : int
- __add_data_to_main_structure(id, name, tf, mf, aua, fd, cf, thread_id) : None
- __get_new_session_for_thread(self, thread_id) : None
- __get_friendship_duration_from_friend_id(self, friend_id, thread_id) : int
- _get_FB_user_account_age_as_days(self, soup, thread_id) : int
- __get_friend_friend_list_uri(self, soup) : str
- __get_facebook_user_account_age(self, facebook_user_id, thread_id, is_fs_friend) : int
- __get_facebook_user_amount_of_friends_from_first_friends_page(self, soup) : int
- _get_facebook_user_amount_of_friends(self, facebook_user_friend_list_uri, thread_id) : int
- __get_sc_friend_data(self, tag, connecting_friend_id, thread_id) : None
- __get_soup_to_next_page_of_friends(self, uri_to_next_page_tag, thread_id) : soup
- __crawl_friend_list_of_friends(self, friend_id, soup, page_num, thread_id) : None
- crawl_data_of_user_friends(self, initial_data_filename, thread_id) : None
- initiate_osn_dicts_and_sessions(self, num_of_threads) : None
- arrange_csv_file_from_mid_data(self, path_to_mid_data_jsons_folder, path_to_csv_file) : None
1
MyThread
+ threadID : int
+ name : str
+ paths : [str]
+ run()

Algorithm solver and Graph Builder class diagram:



5.2 - Class description

5.2.1 src.BL.Graph

- Class name: GraphBuilder
    - Responsibilities: Creates the graph that is going to be presented to the user. Receives input stream from AlgorithmSolver class and creates list of nodes and edges.
    - Main fields:
        - graph: NetworkX - the library instance that handles the graphic subject.
    - Main methods:
        - void draw():
            - Description: This methods draws the created graph and presented it to the screen.

5.2.2 src.BL.AlgorithmSolver

Class name: AlgorithmSolver

- Responsibilities: receives CSV as input, process the CSV file and creates Node for each user; each connection between user is represented by an Edge.
  Also, this class responsible to implement the theoretical part of our project; In this class we apply the theory described by our advisors' articles.
- Main fields:
    - nodes: List<Nodes> - list of nodes
    - edges: List<Edges> - list of edges
- Main methods:
    - void generate():
        - Description: the main method of this class, this method processes and generates all the data needed to build the output graph. Analyses the CSV file that has been received.
    - create_nodes_and_edges():
        - Description: this method called from generate method and creates nodes and edges from the CSV file.
          It updates the nodes and edges lists.
    - calculate_weights():
        - Description: this method calculates weight for each node and edge, the 'weight' field is updated for each object.
    - calculate_TSP():
        - Description: in this method the TSP field is calculated for each Node according to the formula described in the article.

Class name: Edge

- Responsibilities: edge object that is a main component of the graph and represents connection between two nodes.
- Main fields:

        - src: Node - source node of specific edge
        - Dest: Node - destination node of specific edge
        - fd: int - friendship duration between two nodes
        - weight: int - each edge has his own weight based on the formula calculation

Class name: Node

- Responsibilities: node object that is a main component of the graph and represents connection a single user in the social network.
- Main fields:
    - Idd: str - string that contains the id of each user
    - name: str - the name of each user
    - tf: str - this user’s total number of friends
    - mf: str - number of mutual friends between this user and the ego user
    - aua: str - age of user account, when the user first became member of the social network.
    - fd: str - friendship duration between this user and the ego user
    - cf: list<str> - list of user’s mutual friends ids between this user and the ego user of the system
    - weight: int - each node has his own weight according to the formula
    - tsp: int - each node has his own TSP number, calculated according to the article formula.
- Main methods:
    - Node(list<int>..):
        - Description: constructor for the node object, receives as a param the data for creating the node.
    - setWeight(int) :
        - Description: setter for the weight field of each node.

5.2.3 src.BL.FBCrawler

Class name: FBCrawler

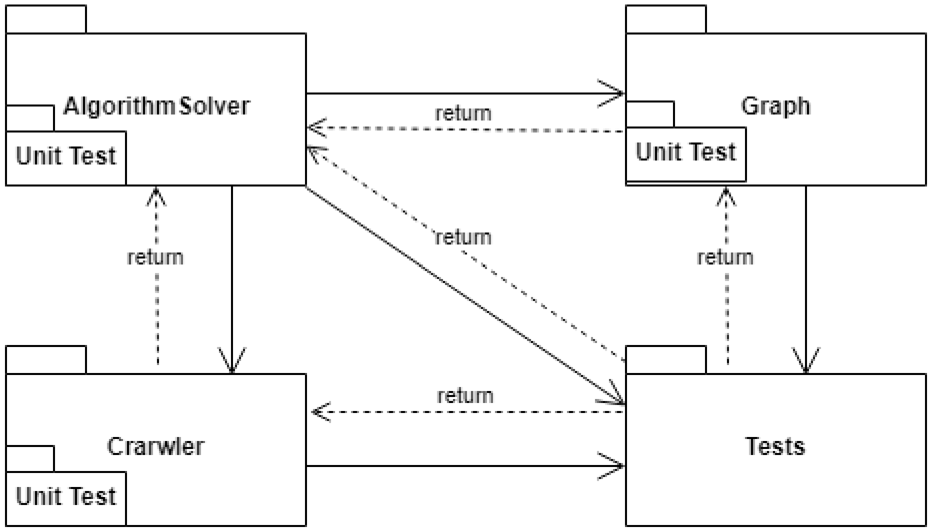
- Responsibilities: one of the main components of our system.
  This class is responsible for scanning the social network, gathering the necessary data that our system needs and creating CSV file contains the entire data that were collected from the internet.
- Main fields:
    - osn_data: dict - holds the data that is going to be inserted to the CSV file.
    - exception_thrown: list - list of exception that were received, the crawler is responsible to handle each exception in the proper way.
- Main methods:
    - void : crawl_data_of_user_friends(initial_data_filename, thread_id):
        - Description: crawl data of first circle friends of user
    - void : get_sc_friend_data(self,tag,connecting_friend_id,thread_id):
        - Description: crawl data of second circle friends of user

5.2.3 src.BL.myThread
Class name: myThread

- Responsibilities: Running crawler on all user’s friends is a large task to execute in serial execution. Therefore, this class will receive a part from the complete task and will execute simultaneously to other threads
- Main fields:
    - facebook_crawler: FBCrawler- The Crawler that unique to the thread.
- Main methods:
    - void : run():
        - Description: execute part of the crawling task

5.3 - Packages

The following package diagram illustrates the main packages and interaction between them in our system.



<u>AlgorithmSolver</u>
The AlgorithmSolver package responsible to all of the calculation operations, including process the CSV file, creates Nodes and Edges, calculate weights for every Node,Edge and TSP for each Node at the second circle.

<u>Crawler</u>
The Crawler package responsible to scan the social network and collect the data of user into CSV file.

<u>Graph</u>
The Graph package responsible to create the visual graphs.

<u>Tests</u>
The Test package responsible to test each package and make ensure that the system is well.

5.4 - Unit Tests

Crawler Package:

1. Test log in to Facebook -
   The main idea is to validate that the system will represent a message that fit the result of log in action.
   If the user insert a valid credentials the system will allow him to login successfully to Facebook.
   If the user enter non valid email or password that not fit the email, The system will present the user a message that the login action failed.
2. Test friendship duration -
   This test tests the capability of the FBCrawl module to extract the friendship duration from facebook timeline page.
   There are user that block the capability of other users to watch their friendship duration and the system needs to return default value that represent that the value is blocked.
3. Test Facebook user age -
   This test tests the capability of the FBCrawl module to extract the age of the Facebook user since he joined to Facebook.
   There are users that have facebook's post from the first year since they joined and there are user that have only the year that they joined to facebook - The test should test generating user age in form of days.
4. Test amount of friends -
   This test tests the capability of the FBCrawl module to extract the amount of friends that Facebook user have.
   This test should cover the scenarios of non public data in facebook user account.

Algorithm Solver Package:

1. Test decode non english names -
   For many facebook users, their names are not in english. And saving those word in CSV file require encoding.
   This test make sure that decoding function revert the encoded name to the form that they were represented in Facebook.
2. Test creation of Graph -
   This test make sure that the graph building process is building the graph from CSV file correctly.

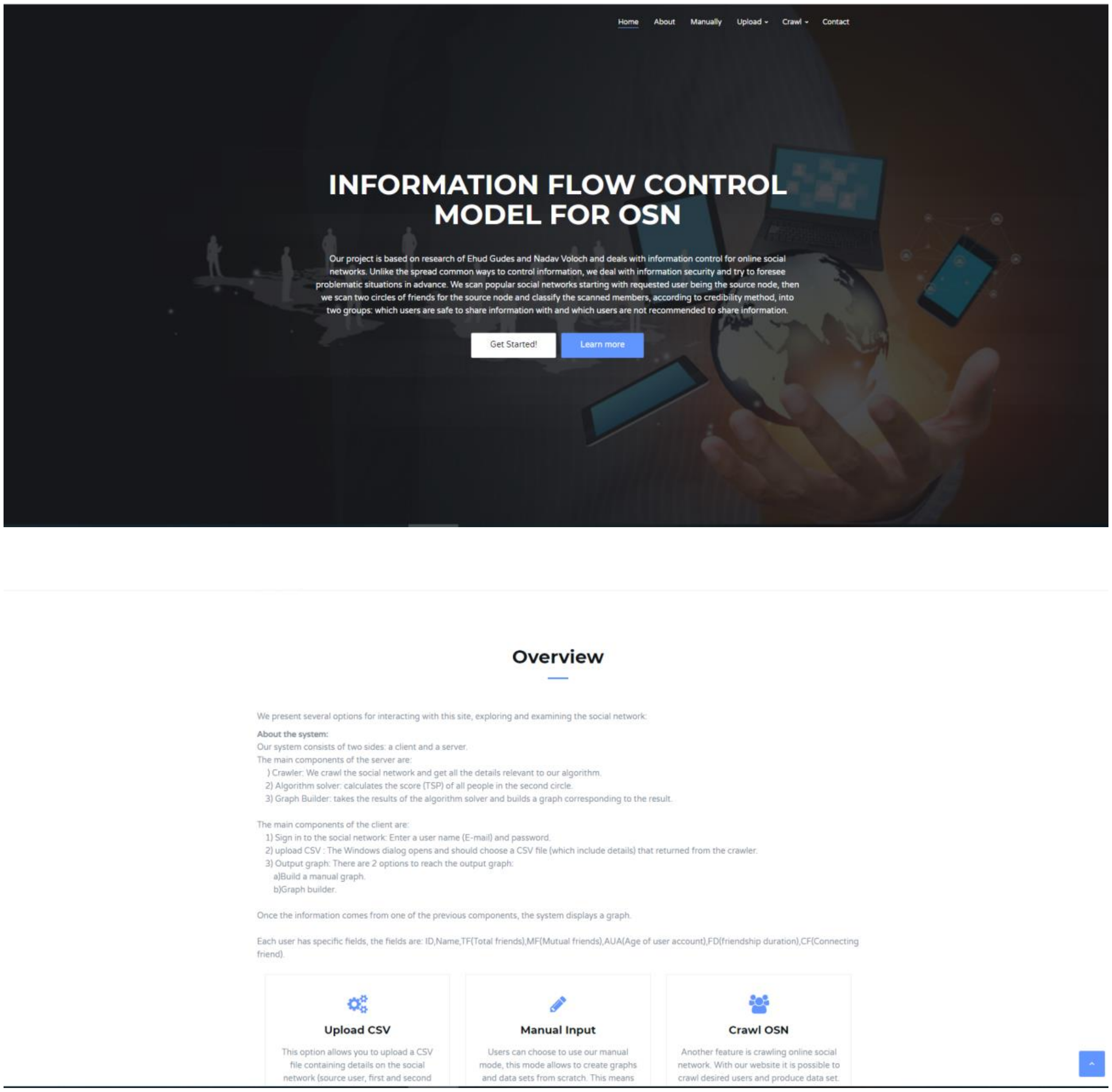
Make sure that all the nodes and the edges were generated properly, and for each node or edge the internal data is same as the csv generating.

3. <u>Test Weight Calculation</u> -
   This test should be based on the main algorithm that represent in the main article and make sure that the weight of each node and edge in the graph after running the algorithm is as expected.
4. <u>Test TSP calculation</u> -
   This test should be based on the main algorithm that represent in the main article and make sure that the credibility score of friends in second circle is as expected.

Chapter 6

User Interface Draft

Start screen



INFORMATION FLOW CONTROL MODEL FOR OSN

Our project is based on research of Ehud Gudes and Nadav Voloch and deals with information control for online social networks. Unlike the spread common ways to control information, we deal with information security and try to foresee problematic situations in advance. We scan popular social networks starting with requested user being the source node, then we scan two circles of friends for the source node and classify the scanned members, according to credibility method, into two groups: which users are safe to share information with and which users are not recommended to share information.

Overview

We present several options for interacting with this site, exploring and examining the social network:

About the system:
Our system consists of two sides: a client and a server.
The main components of the server are:
) Crawler: We crawl the social network and get all the details relevant to our algorithm.
2) Algorithm solver: calculates the score (TSP) of all people in the second circle.
3) Graph Builder: takes the results of the algorithm solver and builds a graph corresponding to the result.

The main components of the client are:
1) Sign in to the social network: Enter a user name (E-mail) and password.
2) upload CSV : The Windows dialog opens and should choose a CSV file (which include details) that returned from the crawler.
3) Output graph: There are 2 options to reach the output graph:
a)Build a manual graph.
b)Graph builder.

Once the information comes from one of the previous components, the system displays a graph.

Each user has specific fields, the fields are: ID,Name,TF(Total friends),MF(Mutual friends),AUA(Age of user account),FD(friendship duration),CF(Connecting friend).

Upload CSV

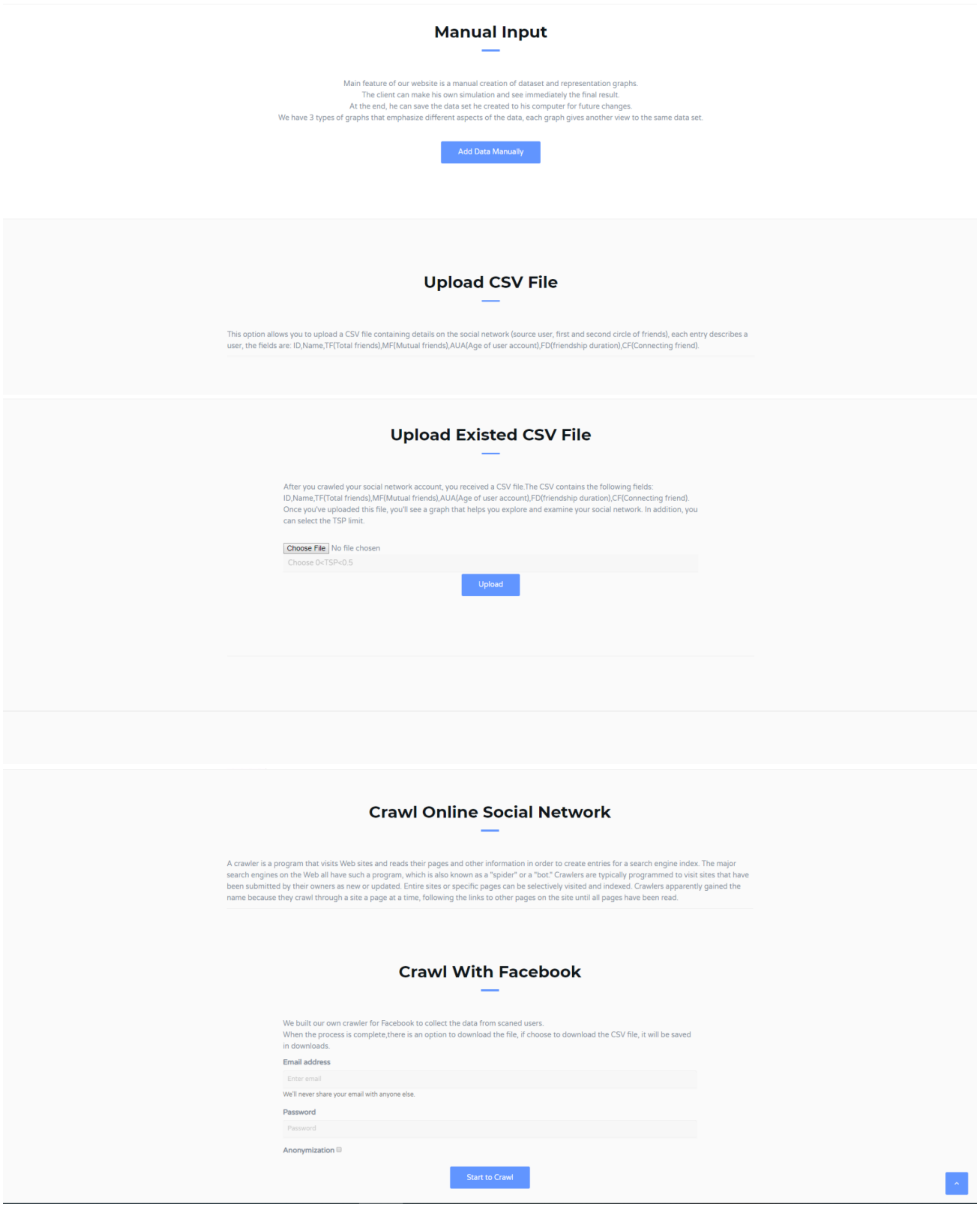
This option allows you to upload a CSV file containing details on the social network (source user, first and second

Manual Input

Users can choose to use our manual mode, this mode allows to create graphs and data sets from scratch. This means

Crawl OSN

Another feature is crawling online social network. With our website it is possible to crawl desired users and produce data set.



Manual Input

Main feature of our website is a manual creation of dataset and representation graphs.
The client can make his own simulation and see immediately the final result.
At the end, he can save the data set he created to his computer for future changes.
We have 3 types of graphs that emphasize different aspects of the data, each graph gives another view to the same data set.

Add Data Manually

Upload CSV File

This option allows you to upload a CSV file containing details on the social network (source user, first and second circle of friends), each entry describes a user, the fields are: ID,Name,TF(Total friends),MF(Mutual friends),AUA(Age of user account),FD(friendship duration),CF(Connecting friend).

Upload Existed CSV File

After you crawled your social network account, you received a CSV file.The CSV contains the following fields: ID,Name,TF(Total friends),MF(Mutual friends),AUA(Age of user account),FD(friendship duration),CF(Connecting friend). Once you've uploaded this file, you'll see a graph that helps you explore and examine your social network. In addition, you can select the TSP limit.

Choose File No file chosen

Choose 0<TSP<0.5

Upload

Crawl Online Social Network

A crawler is a program that visits Web sites and reads their pages and other information in order to create entries for a search engine index. The major search engines on the Web all have such a program, which is also known as a "spider" or a "bot." Crawlers are typically programmed to visit sites that have been submitted by their owners as new or updated. Entire sites or specific pages can be selectively visited and indexed. Crawlers apparently gained the name because they crawl through a site a page at a time, following the links to other pages on the site until all pages have been read.

Crawl With Facebook

We built our own crawler for Facebook to collect the data from scaned users.
When the process is complete,there is an option to download the file, if choose to download the CSV file, it will be saved in downloads.

Email address

Enter email

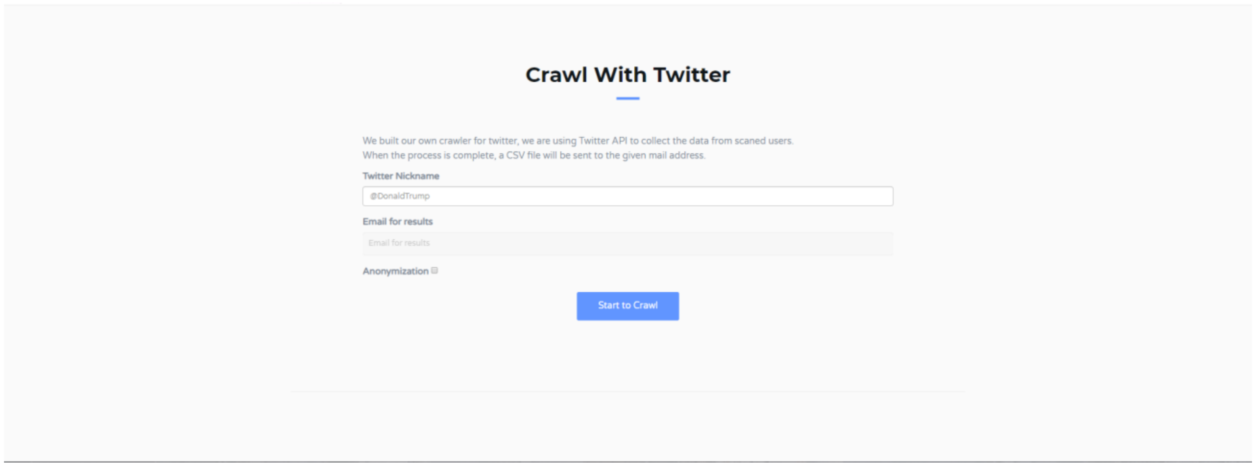
We'll never share your email with anyone else.

Password

Password

Anonymization ☐

Start to Crawl



Crawl With Twitter

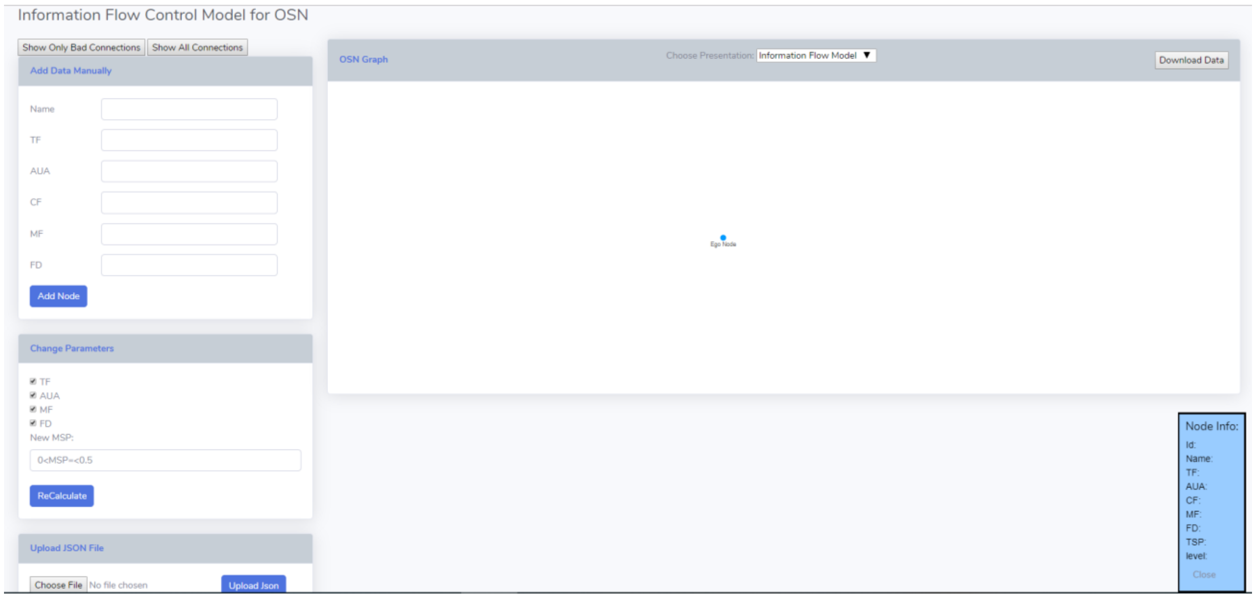
We built our own crawler for twitter, we are using Twitter API to collect the data from scaned users.
When the process is complete, a CSV file will be sent to the given mail address.

Twitter Nickname: @DonaldTrump

Email for results: Email for results

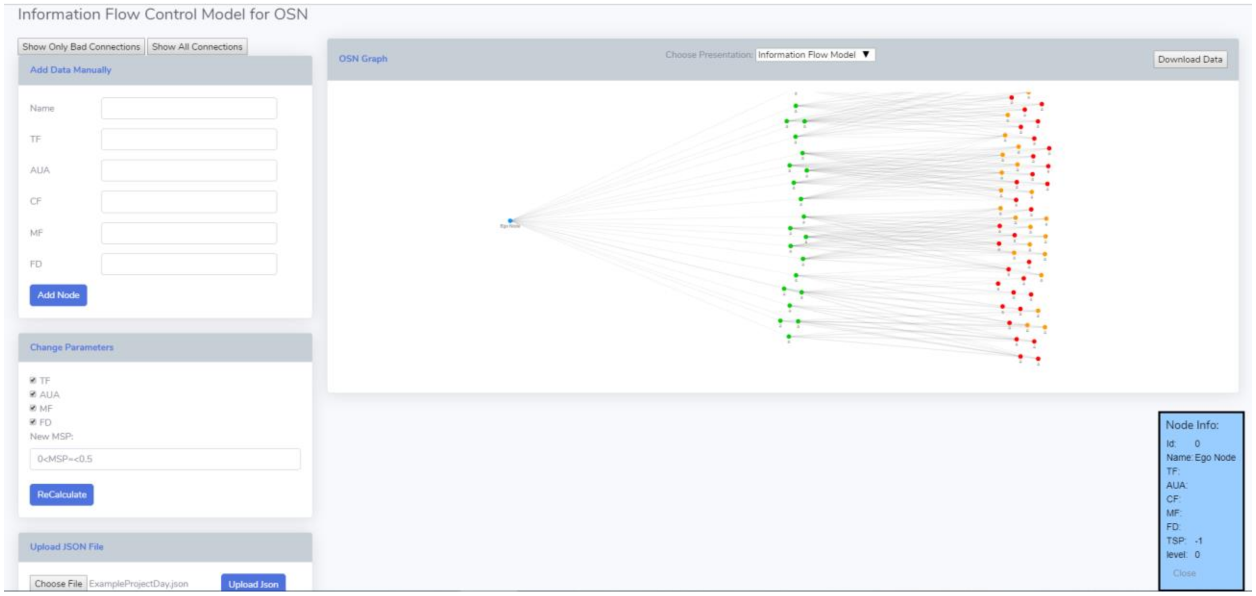
Anonymization ☐

Add manually data screen



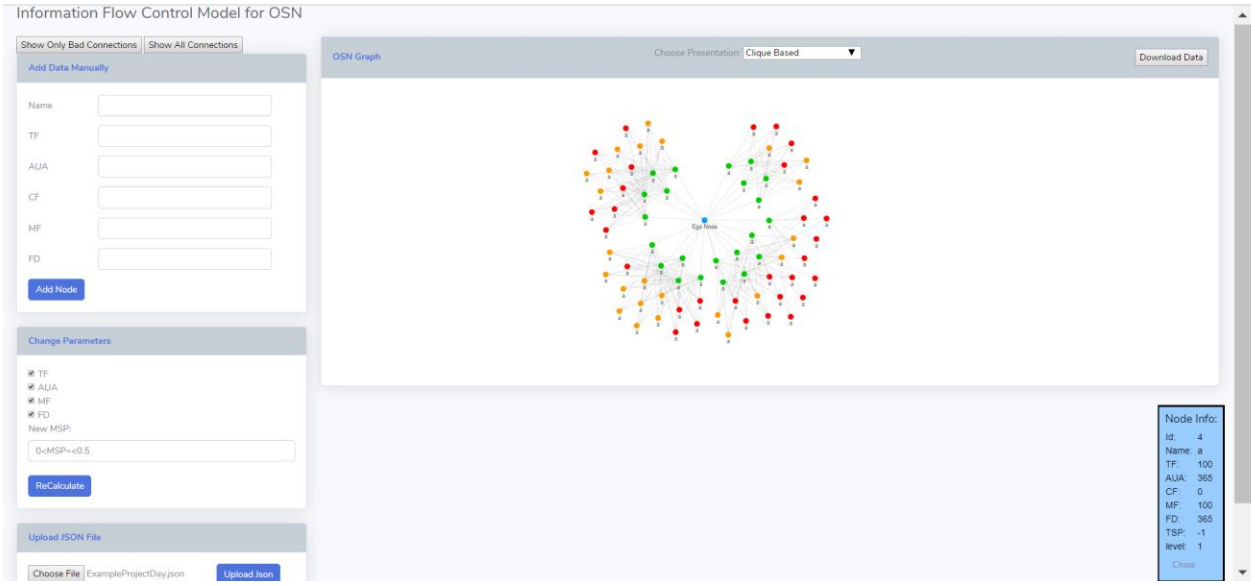
Information Flow Control Model for OSN

<u>Information flow graph screen</u>



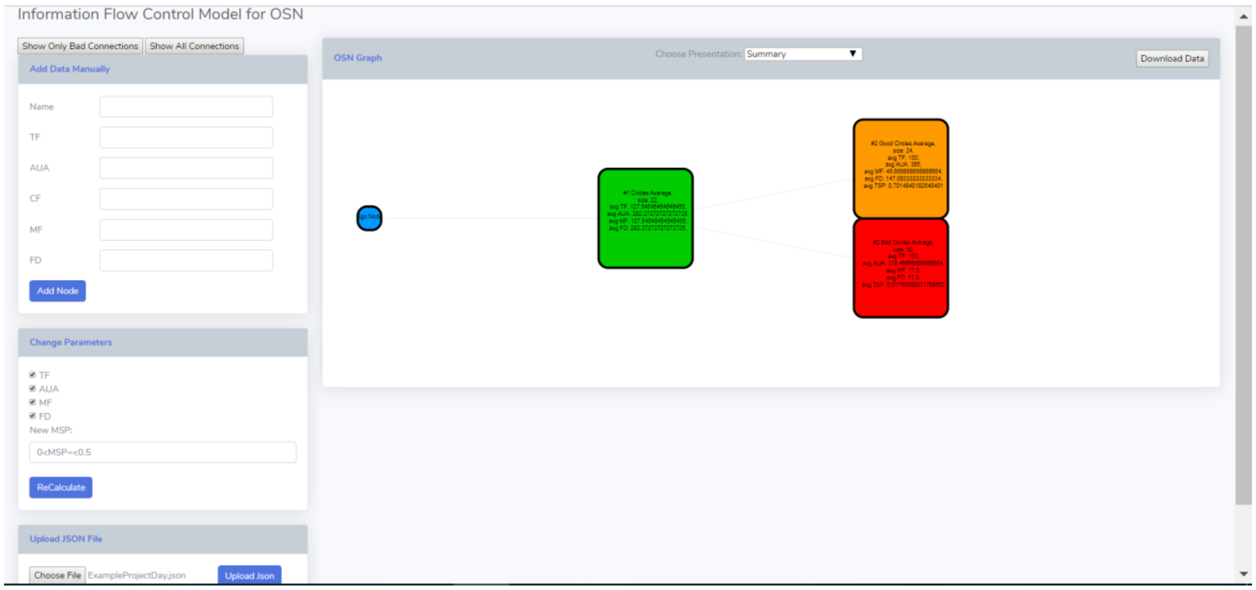
Node Info:
Id: 0
Name: Ego Node
TF:
AUA:
CF:
MF:
FD:
TSP: -1
level: 0

<u>Clique based graph screen</u>



Node Info:
Id: 4
Name: a
TF: 100
AUA: 365
CF: 0
MF: 100
FD: 365
TSP: -1
level: 1

Summary Graph screen


Information Flow Control Model for OSN

ExampleProjectDay.json

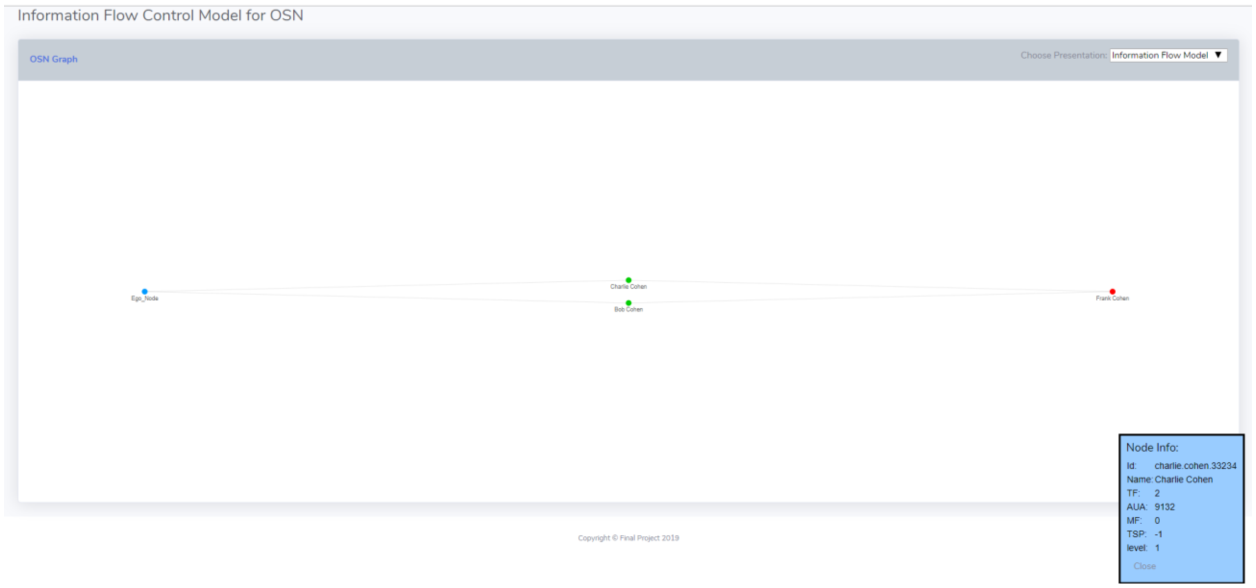
Choose Presentation: Summary

#1 Circles Avarage,
size: 22,
avg TF: 127.54545454545455,
avg AUA: 282.27272727272725,
avg MF: 127.54545454545455,
avg FD: 282.27272727272725,

#2 Good Circles Avarage,
size: 24,
avg TF: 100,
avg AUA: 365,
avg MF: 46.666666666666664,
avg FD: 147.08333333333334,
avg TSP: 0.7014840182648401

#2 Bad Circles Avarage,
size: 30,
avg TF: 100,
avg AUA: 316.46666666666664,
avg MF: 17.5,
avg FD: 17.5,
avg TSP: 0.017603000717996662

Upload CSV file screen


Information Flow Control Model for OSN

Choose Presentation: Information Flow Model

Node Info:
Id: charlie.cohen.33234
Name: Charlie Cohen
TF: 2
AUA: 9132
MF: 0
TSP: -1
level: 1

Copyright © Final Project 2019

Chapter 7

Testing Plan

Functional requirements:

1. The system will run steadily on a deployed web server.

Description	Input	Expected result
Validate connectivity with server	Request the server to check if it up	Message that confirm the server is up
Validate connectivity with server while disconnect the server	Request the server to check if it up	Message that confirm the server is down

2. The system will allow the user to connect to his social network and to gain an access to his social connection

Description	Input	Expected result
Connecting to Facebook and Twitter with valid email and pass	Valid email and password	Login success
Connecting to Facebook and Twitter with non valid email or pass	Non valid email or password	Login failed

3. The system will support **dynamic** information gathering of logged in user

Description	Input	Expected result
Validate correct amount of friends of user	Url to user page	Amount of friend as seen on Facebook or Twitter
Validate correct mutual friends of user	Url to user page	Mutual friends as seen on Facebook or Twitter
Validate correct age of user	Url to user page	User age as seen on Facebook or Twitter
Validate friendship duration of user	Url to user page	friendship duration as seen on Facebook or Twitter

4. The system will process the gathered information

Description	Input	Expected result
Validate CSV file correctness	CSV file	All data in CSV file is arranged and inserted correctly

5. The system will produce a CSV file of the collected information

Description	Input	Expected result
Validate encryption of CSV file and check anonymization	CSV file	All data in CSV file is encrypted and anonymized

6. The system will support **static** upload of already generated CSV file

Description	Input	Expected result
Validation of uploaded CSV file - Non malicious file	CSV file	90% identification of malicious files

7. Implementation of the algorithm described in the main article that our project dealing

Description	Input	Expected result
Check calculation of weight of graph and nodes	Graph	Correctness of weights
Check calculation of weight of TSP of second circle friends	Graph	Correctness of tsp

8. Graph representation to the user

Description	Input	Expected result
Validate represented weights and TSP	Graph	Correct weights and TSP

Graph chosen by the user	Graph	Graph chosen by the user
Graph filtering by the user	Graph	Part of graph that the user choose
Graph zoom in	Graph	Part of graph that the user wanted to zoom in
Graph zoom out	Graph	Part of graph that the user wanted to zoom out
Graph information on each node	Graph	Correct information on node

9. Manually add data

Description	Input	Expected result
Validate Manually data add	Graph with 1 Node	Graph with 2 Nodes
Upload saved JSON file	JSON file	Right graph details.
Save manually new data	New manually graph	New JSON file(with the new data that just added)

Non-Functional requirements:

1. Performance

Description	Input	Expected result
Validate Crawling run time	Url to user page	(end time)-(start time)= O(A*n*m)
Validate algorithm run time	CSV file	(end time)-(start time)= O((V^2)*E)
False negative ratio	CSV file	**|** Adversary connections not discovered **| / |** Adversary connections in total **|**<=0.25
False positive ratio	CSV file	**|** Acquaintance connections calculated as Adversary **| /** **|** Acquaintance connections in total **|** <=0.05

2. Reliability & Stability

Description	Input	Expected result
concurrent users	CSV file	The system will not crashed when 150 or less concurrent users.
Crashes handle	CSV file	The system restart the process when error occur(such as disconnect from server) and doesn’t crashed.
Fill necessary information	Url to user page	All of the fields in the CSV file are full.

3. Safety & Security

Description	Input	Expected result
Permissions to grant to our system	Url to user page	Pop-up window with the permissions he agrees to grant to our system.
Delete data		Our server doesn’t consist any data of users at the end of the process.
Data anonymization	CSV file	CSV file with data after being anonymized

4. Portability

Description	Input	Expected result
Portability	Google Chrome browser	The system interface will run on a Web server.

5. Usability

Description	Input	Expected result
Intuitive system	The entire system	The system is very intuitive and simplify.

6. Availability

Description	Input	Expected result
HTTP requests	Url to user page	As long as the server is running it will wait and respond to HTTP requests.
Social network dysfunctions	CSV	our system will still provide offline support such as CSV import and analyzation, although social network dysfunctions such as social network removed (it possible to check it by import a CSV file that inserted manually).

Integration and Deployment tests:

Description	Input	Expected result
Full success full cycle of crawling from Facebook or Twitter -> generating graph -> running the algorithm -> graph presentation	Full cycle	Successful cycle
Unsuccess cycle of crawling from Facebook or Twitter -> generating graph -> running the algorithm -> graph presentation	Crush during the crawling	Error message during the crawling and stop of process
Unsuccess cycle of crawling from Facebook or Twitter -> generating graph -> running the algorithm -> graph presentation	Crush during the graph generating	Error message during the graph generating and stop of process
Unsuccess cycle of crawling from Facebook or Twitter-> generating graph -> running the algorithm -> graph presentation	Crush during the graph presenting	Error message during the graph presenting and stop of process