Software Design Document

Written by Enna Grigor

Project: Web based tool that identifies dangerous users in social media networks via text and images.

Table of Contents:

1. Introduction

We will start by introducing the product –
The product is web-based tool that identifies dangerous users in social media networks via text and images.

1.1 Purpose

The purpose of the product is to scan social media platforms that have public APIs and extract text and images, then it will monitor and identify keywords according to a pre-defined "dangerous" vocabulary. With the help of both types of monitoring and identifying (text and image) we can maintain a list of suspicious people who may pose a social danger.

1.2 Scope

The tool will scan text and images from social networks that have public API.
The main goal is to focus on popular APIs for example Twitter, Facebook, YouTube. The text extracted can come from comments, posts, published links. During the scan the tool will monitor and identify keywords according to a pre-defined "dangerous" vocabulary.
For example words like massacre, killing spree, murder, guns, etc.
The tool will scan the images posted by users with the help of image detection, it will identify and alert on images that constitute "red lights" such as images with weapons, flags of terrorist organizations, etc.
With the help of both types of monitoring and identifying (text and image) we can maintain a list of suspicious people who may pose a social danger.
Then, the goal is to classify the level of risk of those people on the list.
For example:

- Risk classification of how "dangerous" the person is.
  (red flag, orange flag, yellow flag, green flag)
- Whether the risk is social or political (dangerous to the country).
- How many more "dangerous" friends the same user has.
- In what geographical area the user inhabits.

Another feature is the ability to enter a username/full name of a user, select criteria and search for it according to those criteria and thus check if it poses a social danger.

1.3 Overview

This document provides the architectural design and data design which is broken down to components.
First, we will introduce an overview of the system.
Later we will show the architectural design which include a system diagram, sequence diagrams, class diagram.
Then we will go over the data design and all of the components.
We will show a human interface design and how the system should look like to the user.
Lastly, we will go over the requirements from the SRS document and compare with this document.

1.4 Reference Materiel

https://lucid.app/ - for diagram design.

2. System Overview

The main structure of this system is a client side and web side architecture.
The server side holds all of the logic, processing of data, API calls, etc.
And the client side holds all of the user side needs.
This is done because there is a complete separation of the client side and server side. This contributes to the ability to multitask – one developer can work on client while the other on the server without interrupting one another.

3. System Architecture

3.1 Architectural Design

The system is divided into two parts, the client side and the server side.
The main components are:

The user – divided into two different types – private organization user and security organization user.

The scan – also divided in to two types of scans according to the type of user and the user needs.

Server – stores all of the logic and implementations.

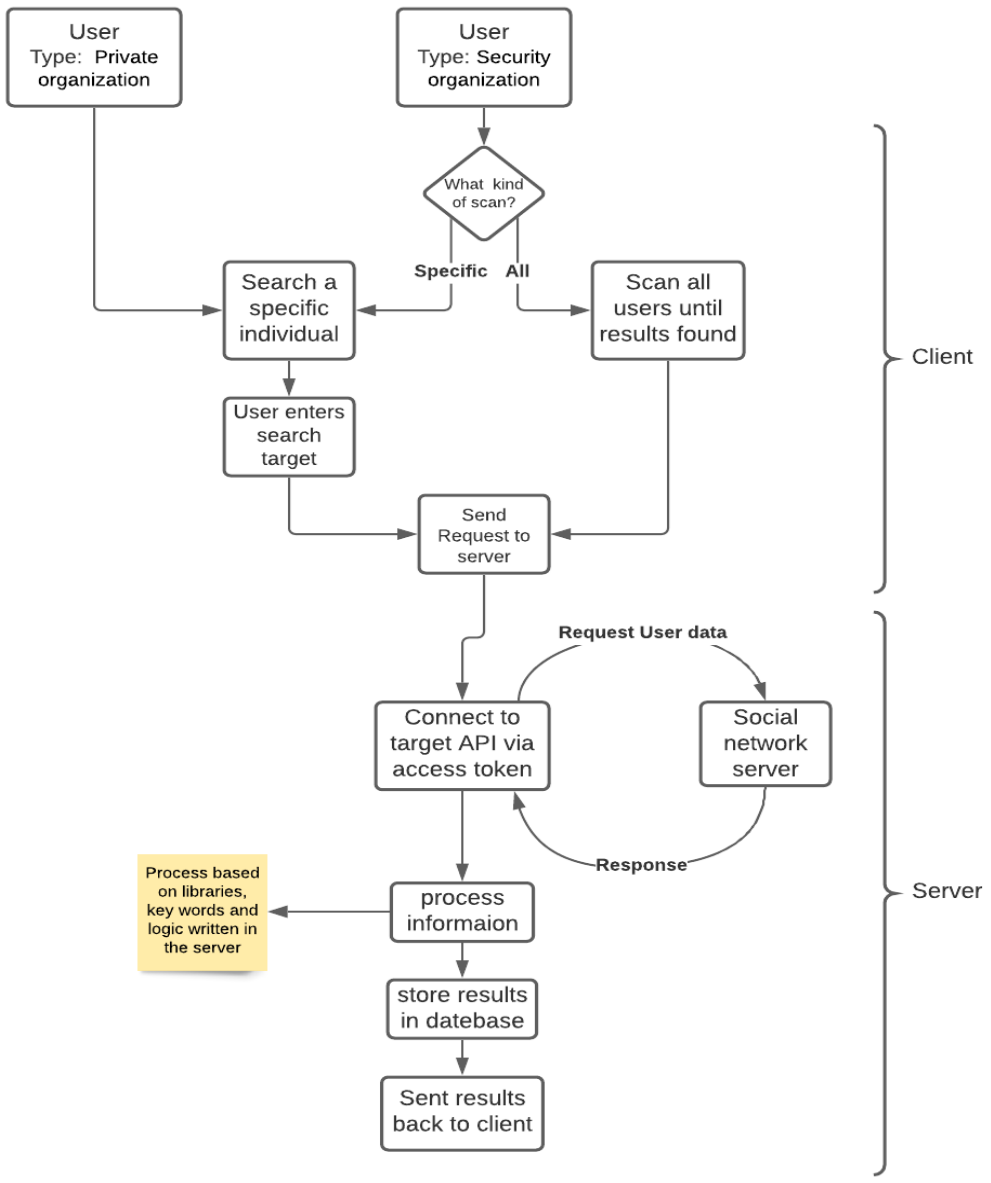
Social media server – lets us get data via request with valid API key.

Database – stores all of the processed information.

As described in the diagram provided below – the interaction can be described as:

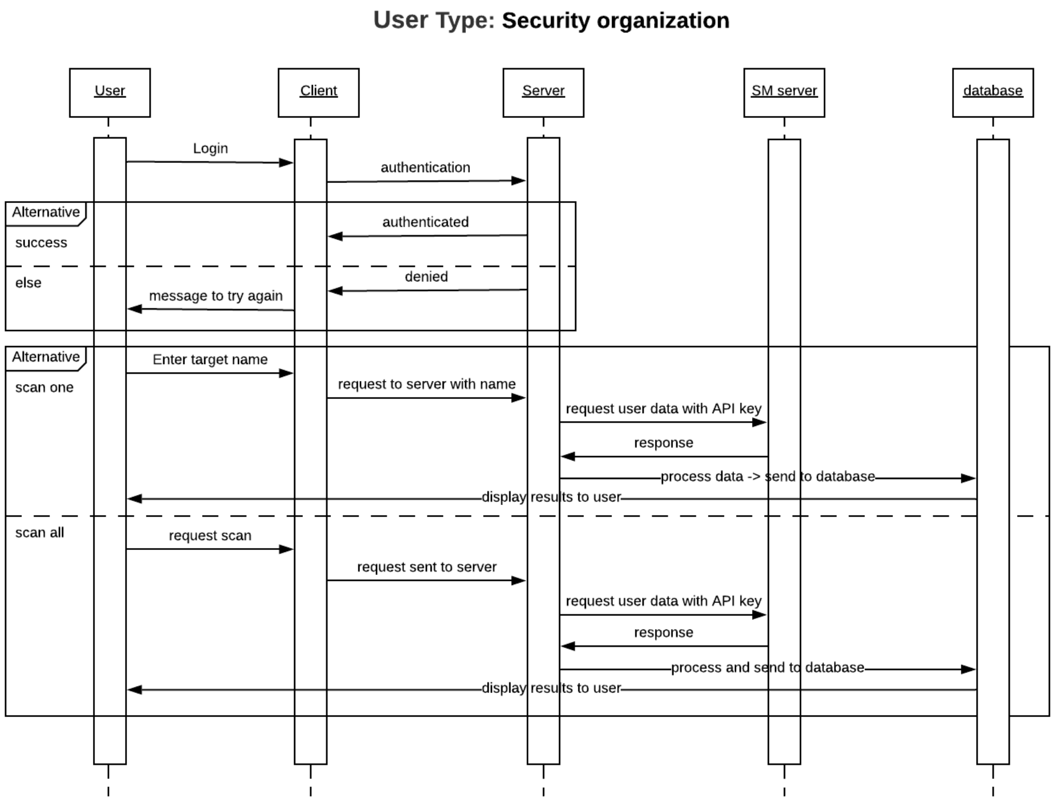
The user sends a request for a scan (the private organization user type can only request a specific user scan) – if the user choose a specific user scan then they have to enter a target name.
After a scan was chosen – the server gets the request.
The server processes the request and sends another request to target social media server with a valid API key.

The social media server responses with the data.
The server then processes the data with the logic written in it and stores the information in the database. After the data is finished processing – the results are sent back to the client.
The client later sees all of the scan results on the website.

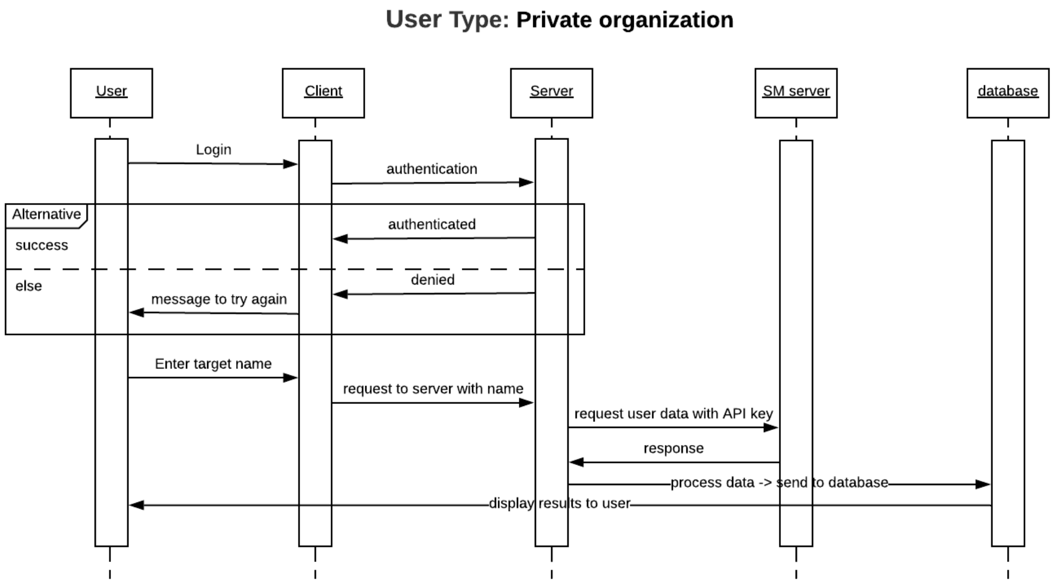

User Type: Private organization
User Type: Security organization
What kind of scan?
Specific
All
Search a specific individual
Scan all users until results found
User enters search target
Send Request to server
Client
Request User data
Connect to target API via access token
Social network server
Response
Process based on libraries, key words and logic written in the server
process informaion
store results in datebase
Sent results back to client
Server

3.2 Decomposition Design

The diagrams provided are two different types of sequence diagrams.
The first is of a security type of user.
Description:
The user tries to log in into their account.
The client sends an authentication request to the server – and the server responses if successful or not. If successful, then the user chooses which scan.
If specific user scan, then they enter a target name – the client sends request to server. The server sends another request to social media server with a valid API key. The social media server returns a response with data. The server processes the data and stores it in the database. The results later are returned to the user.
If scan all option is chooses then the same process is executed but without a target name



For the user of type private organization, the diagram is similar but without the option of scan all users.

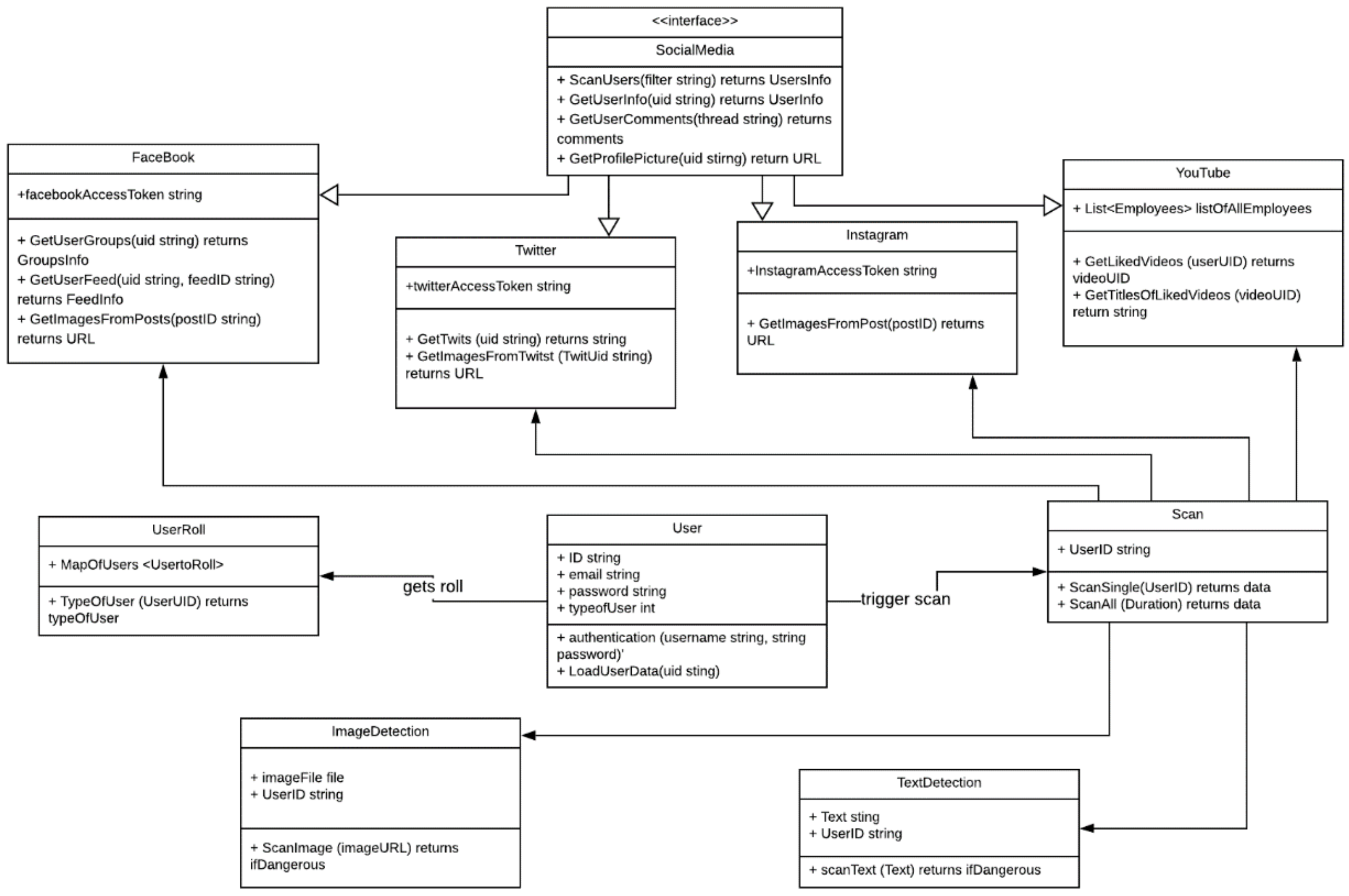


A class/entity diagram is also provided.
The classes provided are:

- ❖ Interface social media:
  Which includes:
  + GetUserInfo(uid string) returns UserInfo
  + GetUserComments(thread string) returns comments
  + GetProfilePicture(uid stirng) return URL
- ❖ Facebook:
  Implements interface social media and also includes:
  + GetUserGroups(uid string) returns GroupsInfo
  + GetUserFeed(uid string, feedID string) returns FeedInfo
  + GetImagesFromPosts(postID string) returns URL
- ❖ Twitter:
  Implements interface social media and also includes:
  + GetTwits (uid string) returns string
  + GetImagesFromTwitst (TwitUid string) returns URL
- ❖ Instagram:
  Implements interface social media and also includes:
  + GetImagesFromPost(postID) returns URL

- YouTube:
  Implements interface social media and also includes:
  + GetLikedVideos (userUID) returns videoUID
  + GetTitlesOfLikedVideos (videoUID) return string
- User:
  The user can ask for user roll and can trigger scan.
  It includes the parameters and following functions:
  + ID string
  + email string
  + password string
  + typeofUser int
  + authentication (username string, string password)
  + LoadUserData(uid sting)
- UserRoll:
  It includes the parameters and following functions:
  + MapOfUsers <UsertoRoll>
  + TypeOfUser (UserUID) returns typeOfUser
- Scan:
  The scan scans the data provided from Facebook, Twitter, Instagram and YouTube.
  It also uses the classes with image and text detection.
  It includes the parameters and following functions:
  + UserID string
  + ScanSingle(UserID) returns data
  + ScanAll (Duration) returns data
- ImageDetection:
  It includes the parameters and following functions:
  + imageFile file
  + UserID string
  + ScanImage (imageURL) returns ifDangerous
- TextDetection:
  It includes the parameters and following functions:
  + Text sting
  + UserID string
  + scanText (Text) returns ifDangerous

Server


<<interface>>
SocialMedia
+ ScanUsers(filter string) returns UsersInfo
+ GetUserInfo(uid string) returns UserInfo
+ GetUserComments(thread string) returns comments
+ GetProfilePicture(uid stirng) return URL
FaceBook
+facebookAccessToken string
+ GetUserGroups(uid string) returns GroupsInfo
+ GetUserFeed(uid string, feedID string) returns FeedInfo
+ GetImagesFromPosts(postID string) returns URL
Twitter
+twitterAccessToken string
+ GetTwits (uid string) returns string
+ GetImagesFromTwitst (TwitUid string) returns URL
Instagram
+InstagramAccessToken string
+ GetImagesFromPost(postID) returns URL
YouTube
+ List<Employees> listOfAllEmployees
+ GetLikedVideos (userUID) returns videoUID
+ GetTitlesOfLikedVideos (videoUID) return string
UserRoll
+ MapOfUsers <UsertoRoll>
+ TypeOfUser (UserUID) returns typeOfUser
gets roll
User
+ ID string
+ email string
+ password string
+ typeofUser int
+ authentication (username string, string password)'
+ LoadUserData(uid sting)
trigger scan
Scan
+ UserID string
+ ScanSingle(UserID) returns data
+ ScanAll (Duration) returns data
ImageDetection
+ imageFile file
+ UserID string
+ ScanImage (imageURL) returns ifDangerous
TextDetection
+ Text sting
+ UserID string
+ scanText (Text) returns ifDangerous

3.3 Design Rational

The rationale for selecting the architecture described in 3.1 is because it gives clear separation between client and server.
It gives an option that more then one developer can work on the system simultaneously. One developer can work on the client side and one can work one the server side without interrupting one another.

4. Data Design

4.1 Data Description

The main data structures that will be used are:

- Nested Map – to list all of the users scanned.
- A key – value data base that will store the information scanned.
- A library of "dangerous" key words that will be stored in the file system.
- A library of "dangerous" images that will be stored in the file system.

4.2 Data Dictionary

- Interface social media:
  Which includes:
  + GetUserInfo(uid string) returns UserInfo
  + GetUserComments(thread string) returns comments
  + GetProfilePicture(uid stirng) return URL
- Facebook:
  Implements interface social media and also includes:
  + GetUserGroups(uid string) returns GroupsInfo
  + GetUserFeed(uid string, feedID string) returns FeedInfo
  + GetImagesFromPosts(postID string) returns URL
- Twitter:
  Implements interface social media and also includes:
  + GetTwits (uid string) returns string
  + GetImagesFromTwitst (TwitUid string) returns URL
- Instagram:
  Implements interface social media and also includes:
  + GetImagesFromPost(postID) returns URL
- YouTube:
  Implements interface social media and also includes:
  + GetLikedVideos (userUID) returns videoUID
  + GetTitlesOfLikedVideos (videoUID) return string
- User:
  The user can ask for user roll and can trigger scan.
  It includes the parameters and following functions:
  + ID string
  + email string

+ password string
+ typeofUser int
+ authentication (username string, string password)
+ LoadUserData(uid sting)

- UserRoll:
  It includes the parameters and following functions:
  + MapOfUsers <UsertoRoll>
  + TypeOfUser (UserUID) returns typeOfUser

- Scan:
  The scan scans the data provided from Facebook, Twitter, Instagram and YouTube.
  It also uses the classes with image and text detection.
  It includes the parameters and following functions:
  + UserID string
  + ScanSingle(UserID) returns data
  + ScanAll (Duration) returns data
- ImageDetection:
  It includes the parameters and following functions:
  + imageFile file
  + UserID string
  + ScanImage (imageURL) returns ifDangerous
- TextDetection:
  It includes the parameters and following functions:
  + Text sting
  + UserID string
  + scanText (Text) returns ifDangerous

5. Component Design

<u>Interface social media:</u>

+ GetUserInfo(uid string) returns UserInfo
Gets user information with an API call.
It may include full name, location listed, email, workplace, gender, age, etc.
+ GetUserComments(thread string) returns comments
Gets user comments with an API call.
It may include comment on posts in pages or groups.
+ GetProfilePicture(uid stirng) return URL
Gets image URL with an API call.

Facebook:
Implements interface social media and also includes:
+ GetUserGroups(uid string) returns GroupsInfo
Gets the groups in which the user is a member off with an API call.
+ GetUserFeed(uid string, feedID string) returns FeedInfo
Gets the feed for the user with an API call.
+ GetImagesFromPosts(postID string) returns URL
Gets the images from posts that where scanned in previous function and saves their URL.

Twitter:
Implements interface social media and also includes:
+ GetTwits (uid string) returns string
Gets all of the twits with API call.
+ GetImagesFromTwitst (TwitUid string) returns URL
Gets all of image URL from twits scanned in previous function.
Instagram:
Implements interface social media and also includes:
+ GetImagesFromPost(postID) returns URL
Gets image URL with API call.

YouTube:
Implements interface social media and also includes:
+ GetLikedVideos (userUID) returns videoUID
Gets a list of liked videos by user via API call.
+ GetTitlesOfLikedVideos (videoUID) return string
Gets the titles of the liked videos via API key.

User:
The user can ask for user roll and can trigger scan.
+ authentication (username string, string password)
The password will be stored via hash in the database.
+ LoadUserData(uid sting)
Will load the type of permission they have (type of user).

UserRoll:
+ MapOfUsers <UsertoRoll>
A nested map of users and what type of user is each one.
+ TypeOfUser (UserUID) returns typeOfUser
Gets type of user from the MapOfUsers.

Scan:
The scan scans the data provided from Facebook, Twitter, Instagram and YouTube.
It also uses the classes with image and text detection.
+ ScanSingle(UserID) returns data
Scans the data that we got from the social media implementation.
+ ScanAll (Duration) returns data
Scans the data that we got from the social media implementation.

ImageDetection:
+ ScanImage (imageURL) returns ifDangerous
Scans images and uses image detection based on a library of pictures provided and determines if image is classified as dangerous or not.

TextDetection:
+ scanText (Text) returns ifDangerous
Scans text and uses text detection based on a library of words provided and determines if the text is classified as dangerous or not.

6. Human Interface Design

6.1 Overview of user Interface

Security organization:

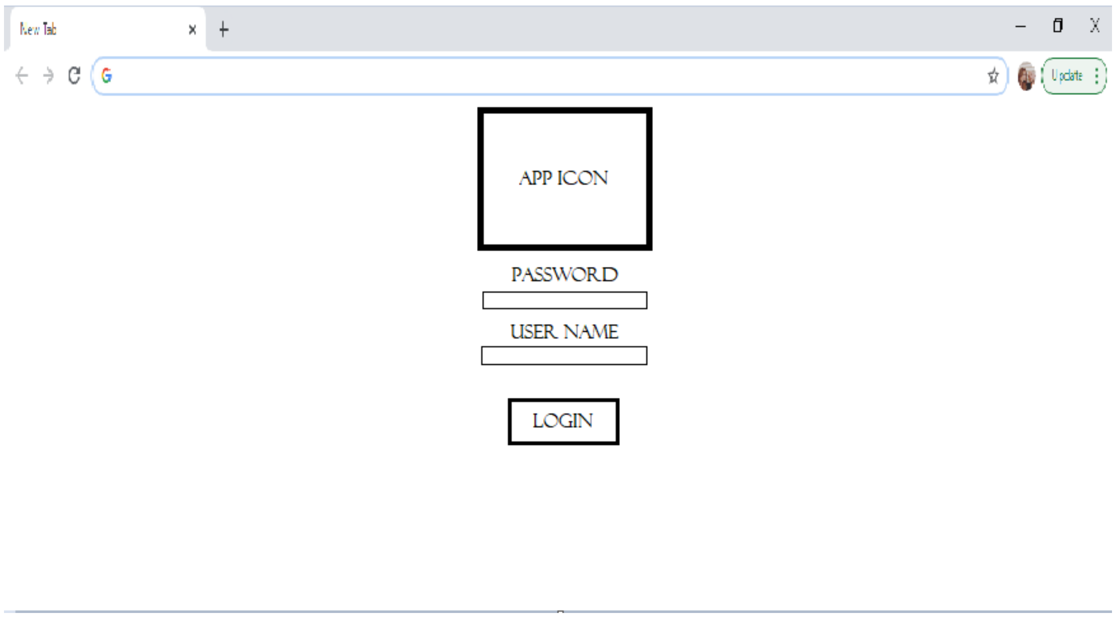
- User friendly layout (the website should be easy to use and data should be displayed in way that could be analyzed easily).
- When entering – should we a login page.
- After login – should display two options of scans.
- The scan should display the source – so if the person who analyzes the data displayed could go to the source and make a final verdict (if the data displayed a false/doubtful classification of the information).
- The data from the scan should be displayed based on the categories that are in natural order (if we are searching for a person – What will we search first?) -
  First the user's name and their risk level, then the links to the social media the user has, the geographical area, then the data that classified this user as "dangerous" – etc.
- The scan should run smoothly with a reasonable time.
- A "report false verdict" button so that the client could report false classification so that the developers could always improve the classification.

Private organizations:

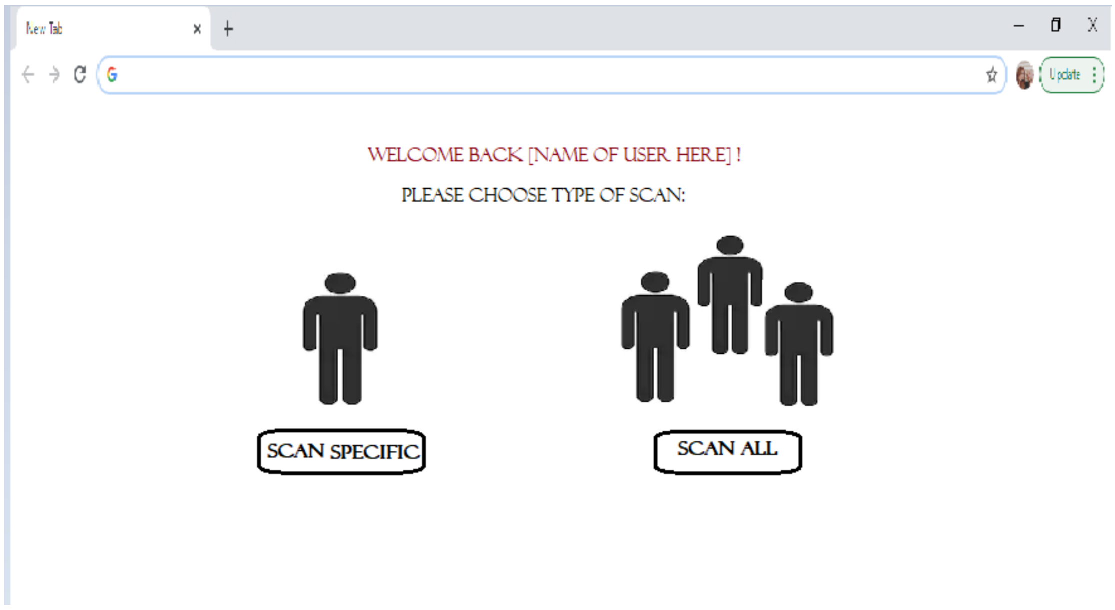
- ❖ When entering – should we a login page.
- ❖ User friendly layout – Obvious place to insert the person's name in order to execute the can.
- ❖ The results should be easy to read – off the bat the user should be classified if dangerous or not (if not – the scan results will simply display – "no dangerous information found – user cleared".
- ❖ The level of risk should be displayed first (if risk is found).
- ❖ If the user is classified as dangerous – then the data that classified the user as "dangerous" should be displayed with links (to make sure that the person who scans - has the opportunity to see why the product made a verdict that user is dangerous).
- ❖ The scan should run smoothly with a reasonable time.
- ❖ A "report false verdict" button so that the client could report false classification so that the developers could always improve the classification

6.2 Screen images

Login page:



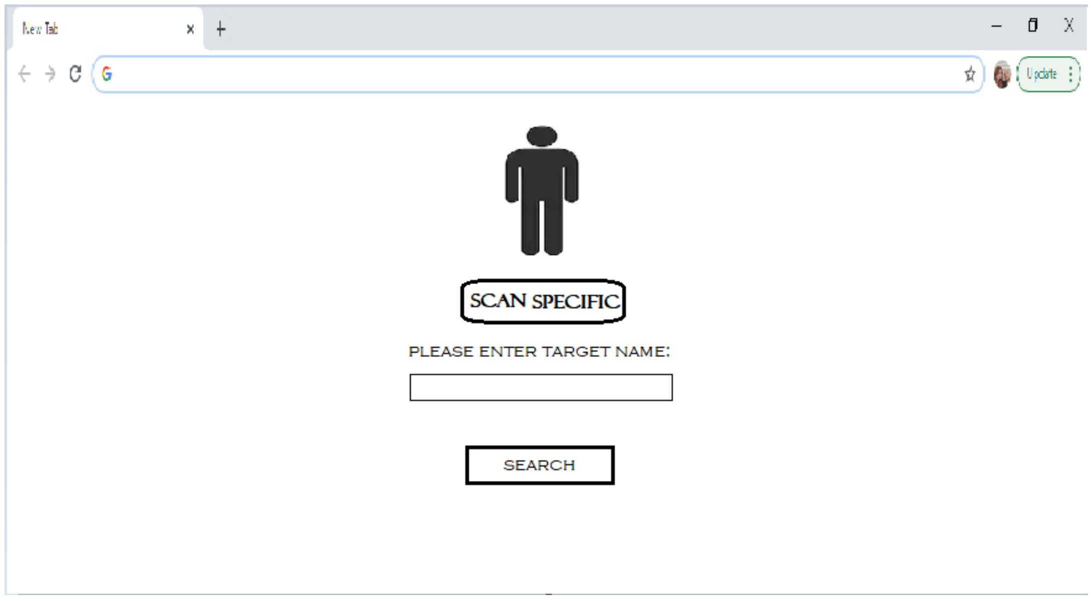
Choose scan:



WELCOME BACK [NAME OF USER HERE] !

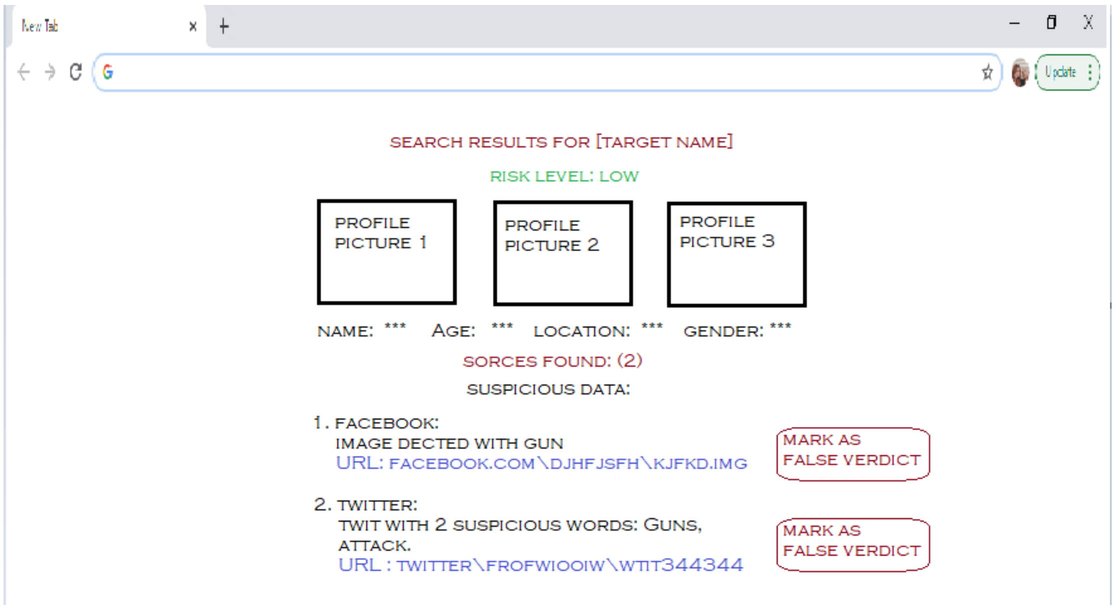
PLEASE CHOOSE TYPE OF SCAN:

Scan specific:



PLEASE ENTER TARGET NAME:

Results for specific:


New Tab

SEARCH RESULTS FOR [TARGET NAME]

RISK LEVEL: LOW

PROFILE PICTURE 1 | PROFILE PICTURE 2 | PROFILE PICTURE 3

NAME: *** AGE: *** LOCATION: *** GENDER: ***

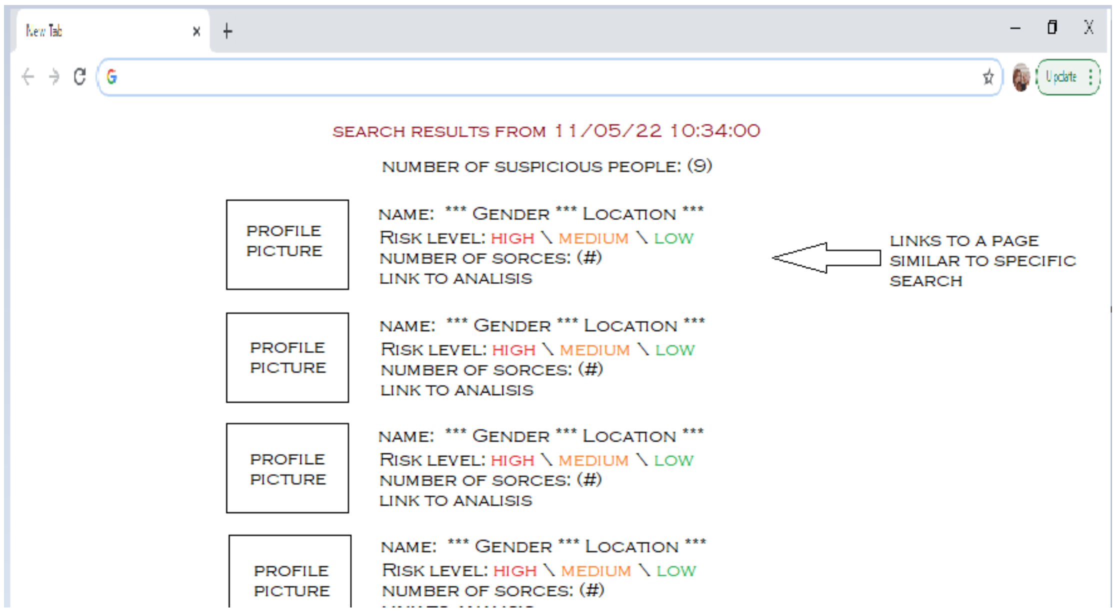
SORCES FOUND: (2)

SUSPICIOUS DATA:

1. FACEBOOK:
   IMAGE DECTED WITH GUN
   URL: FACEBOOK.COM\DJHFJSFH\KJFKD.IMG
   MARK AS FALSE VERDICT

2. TWITTER:
   TWIT WITH 2 SUSPICIOUS WORDS: GUNS, ATTACK.
   URL : TWITTER\FROFWIOOIW\WTIT344344
   MARK AS FALSE VERDICT

Scan all results:


New Tab

SEARCH RESULTS FROM 11/05/22 10:34:00

NUMBER OF SUSPICIOUS PEOPLE: (9)

PROFILE PICTURE
NAME: *** GENDER *** LOCATION ***
RISK LEVEL: HIGH \ MEDIUM \ LOW
NUMBER OF SORCES: (#)
LINK TO ANALISIS

LINKS TO A PAGE SIMILAR TO SPECIFIC SEARCH

PROFILE PICTURE
NAME: *** GENDER *** LOCATION ***
RISK LEVEL: HIGH \ MEDIUM \ LOW
NUMBER OF SORCES: (#)
LINK TO ANALISIS

PROFILE PICTURE
NAME: *** GENDER *** LOCATION ***
RISK LEVEL: HIGH \ MEDIUM \ LOW
NUMBER OF SORCES: (#)
LINK TO ANALISIS

PROFILE PICTURE
NAME: *** GENDER *** LOCATION ***
RISK LEVEL: HIGH \ MEDIUM \ LOW
NUMBER OF SORCES: (#)

6.3 Screen Objects and Actions

Image 1:
Standard login page.
Image 2:
Only for security type user –
Selecting which scan to execute.
Each button when pressed leads to desired landing page.
Image 3:
The user needs to type target input in order to scan – when pressed – "search" data sent to server.
Image 4:
Results for specific search.
Data that was found displayed for user to see with links.
Image 5:
Results from scan shown. Each person found can be further analyzed with a specific search like in photo 4.

7. Requirements Matrix

The product will scan text and images from social networks that have public API.	**Data Design**
The text extracted will come from comments, posts, published links.	**Data Design**
During the scan the product will monitor and identify keywords according to a pre-defined "dangerous" vocabulary.	**Component Design**
With the help of both types of monitoring and identifying (text and image) the product will maintain a list of suspicious people who may pose a social danger.	**Component Design**
Risk classification will be classified based on how "dangerous" the person is. (red flag, orange flag, yellow flag, green flag).	**Screen Images** **Component Design**
The product will display whether the risk is social or political (dangerous to the country).	**Screen Images**
The product will display a list of other dangerous "friends" the user has (if found).	**Screen Images**

The product will display the geographical area the user is active on.	**Screen Images**
The product will allow the user to scan a specific name they insert in order to see if they have "dangerous" behavior.	**Screen Images** **Component Design**
The information extracted will be displayed in a user-friendly way so that it could be easily analyzed (via graphs, maps, links, etc.).	**Screen Images** **Component Design**
The system will allow a "false verdict" report that can be sent to the developers so it could always be improved.	**Screen Images** **Component Design**
The product will be on web-based platform.	**Screen Images**
Only registered users can use product after login authentication.	**Screen Images** **Component Design**
The client side will be written in react.	**Not specified in design**
The server side will be written in GO.	**Not specified in design**
The database that will be used to store the information will be a key-value based database.	**Data description**
The website should load in 3 seconds when the number of simultaneous users is > 10000.	**Not specified in design**
When scanning a specific user – it should not take more than 5 minutes.	**Not specified in design**
When scanning the social media – for dangerous users – the scan will be executed until told otherwise.	**Not specified in design**
The product will send an error while scanning if the user has unstable internet connection or an error occurred.	**Not specified in design**
Each successful login will be recorded in an audit trail.	**Not specified in design**