Project Title:	**Sensing and predictive treatment of frailty and associated co-morbidities using advanced personalized models and advanced interventions**
Contract No:	690140
Instrument:	Collaborative Project
Call identifier:	H2020-PHC-2014-2015
Topic:	PHC-21-2015: Advancing active and healthy ageing with ICT: Early risk detection and intervention
Start of project:	1 January 2016
Duration:	36 months

Deliverable No: D4.8

LingTester Test Results – Active (on-line) mode

Due date of deliverable:	M18 (30th June 2017)
Actual submission date:	29th June 2017
Version:	1
Date:	29th June, 2017
Lead Author:	UoP Makrhs Christos
Lead partners:	UoP

Horizon 2020
European Union funding
for Research & Innovation

CHANGE HISTORY

Ver.	Date	Status	Author (Beneficiary)	Description
0.1	01/04/2017	draft	C. Tsimpouris (UoP), N. Fazakis (UoP)	Initial draft
0.2	01/05/2017	draft	C. Tsimpouris (UoP), N. Fazakis (UoP), C. Makrhs (UoP)	First draft deliverable report
0.3	12/06/2017	draft	C. Tsimpouris (UoP), N. Fazakis (UoP), C. Makrhs (UoP)	Updated deliverable report sent for internal review
0.4	22/06/2017	draft	C. Tsimpouris (UoP), N. Fazakis (UoP),	Second draft deliverable report.
0.5	23/06/2017	draft	I. Kalamaras (CERTH), A. Vasilakis (CERTH)	Revision of the document
0.6	29/06/2017	final	C. Tsimpouris (UoP), N. Fazakis (UoP), C. Makrhs (UoP)	Deliverable finalised taking into account internal review’s comments.

EXECUTIVE SUMMARY

LingTester is the FrailSafe language analysis tool that aims to process the user's typed text and detect abnormal behaviour. At this point, the prototype is in early alpha stage, but still it is able to perform classification according to levels of frailty. The present deliverable describes the development of the online mode of this tool, which covers all steps needed to support all necessary user actions while also removing any sensitive information, and thus, protecting participants' data.

This deliverable is part of WP4. The main objective of this Work Package is to handle the collection, management and analysis of frailty older people data streamed through their social, behavioural, cognitive and physical activities. Offline mode is provided through deliverable D4.10. LingTester online mode wraps the passive model through an API for easy access, while also a web tool provides users the ability to subscribe for this service.

DOCUMENT INFORMATION

Contract Number:	H2020-PHC–690140	**Acronym:**	FRAILSAFE
Full title	Sensing and predictive treatment of frailty and associated co-morbidities using advanced personalized models and advanced interventions		
Project URL	http://frailsafe-project.eu/		
EU Project officer	Mr. Jan Komarek		

Deliverable number:	4.8	**Title:**	LingTester Test Results – Active (on-line) mode
Work package number:	4	**Title:**	Data Management and Analytics

Date of delivery	**Contractual**	30/6/2017 (M18)	**Actual**	29/6/2017
Status	Draft ☐		Final ☒	
Nature	Report ☐	Demonstrator ☒	Other ☐	
Dissemination Level	Public ☒	Consortium ☐		
Abstract (for dissemination)	This deliverable reports on the choices made in the design of the online LingTester system, sub-systems, technical specifications and architecture. Firstly an overall introduction to the system concepts, modules and processes is given; secondly a more detailed presentation of different layers composing the system architecture - devices, frontend interfaces, server backend infrastructure - is presented in all its parts			
Keywords	frailty, frailty classification, natural language processing, API			

Contributing authors (beneficiaries)	Tsimpouris Charalampos (UoP) Fazakis Nikos (UoP) Makrhs Xrhstos (UoP) Megalooikonomou Vasileios (UoP)			
Responsible author(s)	Makrhs Xrhstos		**Email**	makri@ceid.upatras.gr
	Beneficiary	UoP	**Phone**	+30 2610 996968

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ANNEXES

1. INTRODUCTION

The LingTester online mode is constructed based on two main sub modules, frontend and backend, as discussed in Chapter 2. Each submodule is also based on different layers of processes which interact altogether through predefined APIs, existing or custom ones. Chapter 2 describes the architecture of the LingTester online mode, and is a technical introduction of how the system is constructed. Chapter 3 explains the frontend in detail, while Chapter 4 continues to explain how the backend works. Discussion on results is given in Chapter 5, and finally Chapter 6 is an overall summary about legal issues concerning the LingTester online tool.

2. OVERALL ARCHITECTURE

The following image (Figure 1) shows the architecture of the online mode of LingTester, which is discussed in detail within the following chapters. The online mode is based on two main sub modules, frontend and backend which interact through a predefined API within a secure Virtual Private Network (VPN). Users (participants in our case) connect only to the frontend server, as shown in the following figure, and interact with the webservice in a non-intrusive way only to provide access to the third party social networks. The frontend web service is only allowed to interact with the backend, in order to request a new prediction based on the user input, as gathered by the crawler (discussed in detail in Chapter 3.7).

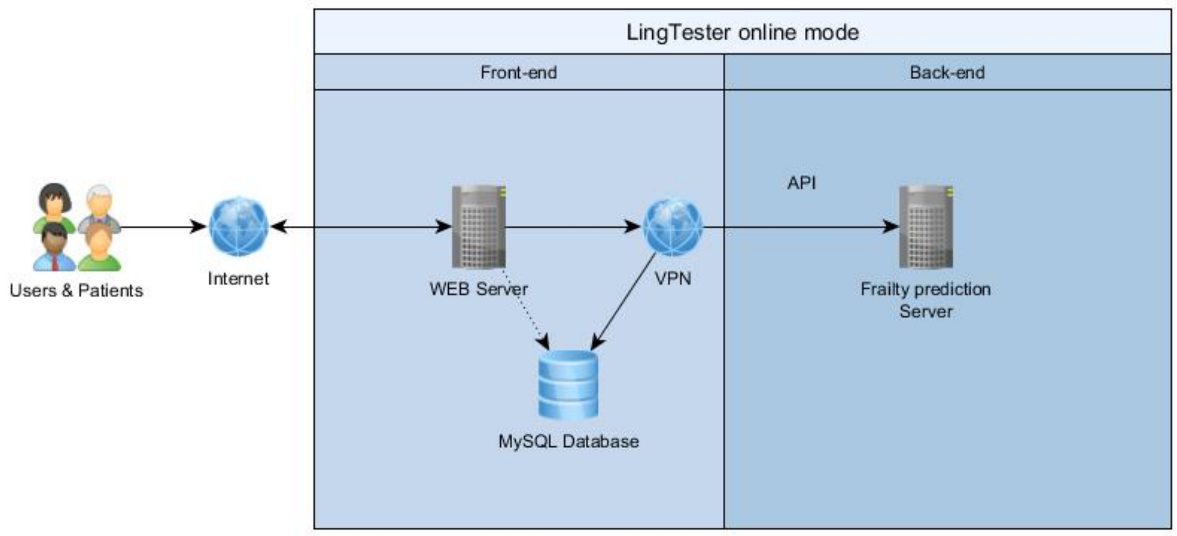

Figure 1: Online mode of LingTester

3. Frontend

The frontend is based on a web server which is publicly available through a known web address. Currently, this URL address has been set to https://lingtester.frailsafe-project.cloud/.

3.1 Architecture

The architecture of the frontend server is a generic LAMP stack, which can be seen in figure 2. LAMP stands for Linux/Apache/MySQL/PHP (Lee & Ware, 2003) and was selected as an ideal option that is easy to maintain as it is based on a vast community where bugs are quickly fixed. In addition, we minimise costs for custom hardware and software due to the fact that this solution can be easily transferred to a new hardware instance for testing or scalability purposes.

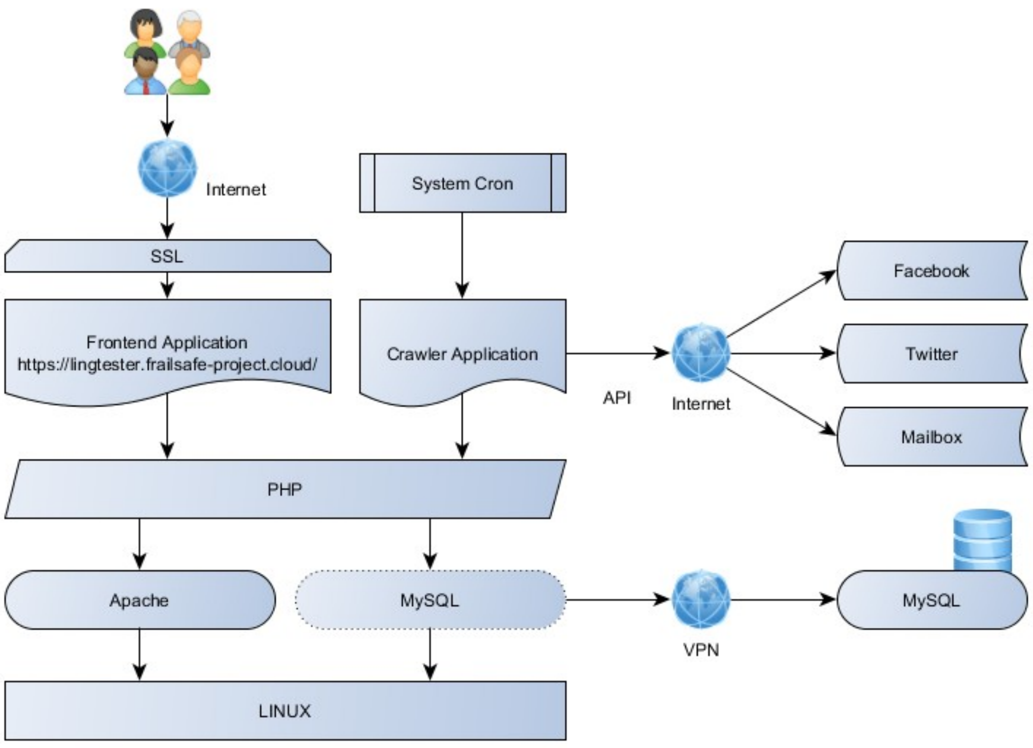

Figure 2: Frontend server, internal architecture

Furthermore, a cron job[1] has been implemented and set to run every minute that initiates the crawler. This crawler is responsible to fetch new data from:

[1] The software utility Cron is a time-based job scheduler in Unix-like computer operating systems. People who set up and maintain software environments use cron to schedule jobs (commands or shell scripts) to run periodically at fixed times, dates, or intervals. It typically automates system maintenance

1. Facebook, for all users that already have authorised access,
2. Twitter, for all users that already have authorised access, and
3. Email submissions, that have been sent at the predefined lingtester mailbox. This mailbox has been created solely for this purpose and is not used for any other purpose

3.2 Installation

In order to install the aforementioned frontend server from scratch the following steps should be reproduced. All commands must be run as root user, in order to overcome any permission hiccups. While all commands are self-explanatory for Linux administrators (Nemeth, 2006), they are followed by a small description.

- `apt-get install apache2`
  - Installs apache necessary binary files

- `sudo apt-get install libapache2-mod-php`
  - Installs php module, to be available within Apache

- `sudo apt-get install mysql-server php7.0-mysql php7.0-mcrypt`
  - Install mysql server and necessary MySQL modules to be available from PHP. While we do not use the local MySQL database, this module is still needed to connect to the remote one

- `wget https://dl.eff.org/certbot-auto`
- `chmod a+x certbot-auto`
- `./certbot-auto`
  - Downloads and executes *certbot* auto, which can auto validate SSL for our local apache server through a user-friendly wizard

- `crontab -e`
  - We should insert the following two cron jobs
  - `0 0 1 */2 * ~/certbot-auto renew --no-self-upgrade`
    - Update currently installed SSL once per two months

  - `* * * * * wget https://lingtester.frailsafe-project.cloud/crawler/  > /dev/null 2>&1`
    - Executes crawler once per minute

- Copy all files needed for the frontend server to run to `/var/www/html/`

---

or administration—though its general-purpose nature makes it useful for things like downloading files from the Internet and downloading email at regular intervals. The origin of the name cron is from the Greek word for time, χρόνος (chronos). Cron is most suitable for scheduling repetitive tasks.

- Import SQL script as given in the Appendix 1 into the available MySQL database for this purpose

- Update all details in the file `config.php` so as to provide access to the MySQL database along with the backend
    - `$dbHos` : the domain name of the MySQL server, which is not necessary to be a local one. This way, we can scale up and support more than one frontend servers, that communicate to the same MySQL server and are all in sync
        - Currently set as **mysql-frailsafe-project-cloud.chpnnoj1sagw.eu-west-1.rds.amazonaws.com:3306**
    - `$dbUsername` : MySQL username
        - Currently set as **uop2**
    - `$dbPassword` : MySQL password
        - Currently set as ************ *(for security purposes this is only provided after request)*
    - `$dbName` : MySQL database
        - Currently set as **lingtester-db**
    - `$backend` : Full URL path to the prediction backend server. Assuming that the backend is only accessible through the Frailsafe VPN, the frontend has already access to the VPN instances.
        - Currently set as **http://172.16.2.131:5000**

3.3 Communication privacy

The frontend server, as it is publicly available, has been protected behind an SSL certificate as also shown in figure 2. SSL[2] (Ristic, 2010) is the backbone of our secure Internet and it protects all sensitive information as it travels across the world's computer networks. SSL is essential for protecting the website, even if it does not handle sensitive information like credit cards. It provides privacy, critical security and data integrity for both the website and the user's (participant’s) personal information.

The primary reason why SSL is used is to keep sensitive information sent across the Internet encrypted so that only the intended recipient can understand it. This is important because the information sent on the Internet is passed from computer to computer to get to the destination FrailSafe server. Any computer between the end-user and the server can view usernames and passwords, and other sensitive information if it is not encrypted with an SSL certificate. When an SSL certificate is used, the information becomes unreadable to everyone except for the server the user is sending the information to. This protects it from hackers and identity thieves.

In addition to encryption, the installed SSL certificate also provides authentication. This means we can be sure that each user is sending information to the right server and not to an

[2] https://en.wikipedia.org/wiki/Transport_Layer_Security

imposter trying to steal this information. This is important, as the nature of the Internet means that any user will often be sending information through several computers/routers. Any of these computers could pretend to be FrailSafe website and trick them into sending them personal sensitive information.

For our server, a valid SSL certificate has been provided by Let's Encrypt[3], a free, automated, and *open* Certificate Authority, brought by the nonprofit Internet Security Research Group (ISRG). As certificates from this authority get auto expired every 3 months, another cron task has been set within the Linux system, to auto renew the certificate without any administration interference.

3.4 Anonymization of data

Our primary concern was to protect participant's data at all times. Therefore, the first step after the initial retrieval of new text from the eCRF was to remove any possibly private data of the following data structures using regular expressions, and therefore we can safely remove the following well defined classes: *credit card numbers*, *emails*, *social security numbers*, *dates of birth*, *zip codes*. No other version of the text, containing private data, was kept. The source code has been constructed in modular way to add more rules, if this becomes necessary.

3.5 MySQL Database schema

An internal database (MYSQL, 2001) has been constructed to support all needed actions, store anonymised data and keep track of history events. The following image shows the EER diagram[4]. Detailed SQL script in order to reconstruct this database is given in Appendix 1.

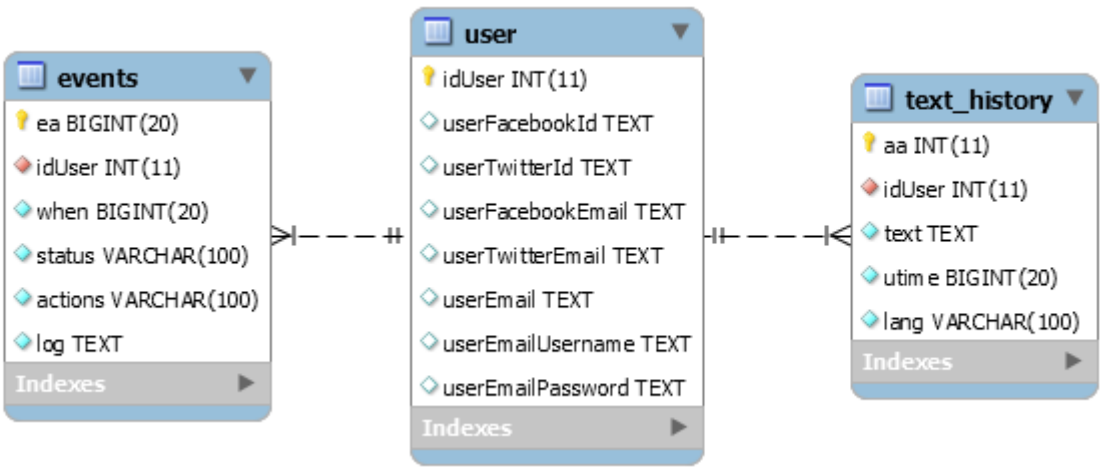

Figure 3. MySQL EER diagram.

[3] https://letsencrypt.org/

[4] The enhanced entity–relationship (EER) model (or extended entity–relationship model) in computer science is a high-level or conceptual data model incorporating extensions to the original entity–relationship (ER) model, used in the design of databases.

Each table is discussed in detail in the following paragraph.

Table **user**: this table stores all users that have concluded the initial process (see chapter 3.6), from which crawler continuously checks for new content.

- *idUser*: user id, auto increment
- *userFacebookId*: Facebook id, in case there is one
- *userTwitterId*: Twitter id, in case there is one
- *userFacebookEmail*: Email, given by Facebook
- *userTwitterEmail*: Email, currently Twitter doesn't provide any information concerning user's email, however, this field has been added for future use
- *userEmail*: Email, as given by the user
- *userEmailUsername*: Email username, in case it is given by the user to access his/her mailbox
- *userEmailPassword*: Email password, in case it is given by the user to access his/her mailbox

Table **text_history**: All crawlable data is saved within this table for future reference

- *aa*: auto increment for each new text
- *idUser*: id of the user, from the previous table, for whom this text has been saved
- *text*: text provided from Facebook, Twitter or mailbox
- *utime*: Unix timestamp[5], of the submission. Unix time (also known as POSIX time or epoch time) is a system for describing instants in time, defined as the number of seconds that have elapsed since 00:00:00 Coordinated Universal Time (UTC), Thursday, 1 January 1970
- *lang*: language detected or provided for this text

Table **events**: For every new text provided through Facebook, Twitter or mailbox, an event is created. This way, we can also step backwards and identify all previous frailty predictions for each user or all users in general

- *ea*: auto increment of the event
- *idUser*: id of the user this event belongs to
- *when*: Unix timestamp of the event
- *status*: frailty prediction
- *actions*: actions taken after this event is triggered. This fields helps to know if an email has been sent to the user. This way, we always know when an email was sent or not, and avoid spamming the user
- *log*: all communication between frontend and backend that was triggered based on this event. This field is only for debugging purposes in order to help technical and administrative personnel identify bugs

3.6 User-flow

[5] https://en.wikipedia.org/wiki/Unix_time

Users are requested to visit the LingTester web server in order to authorise access so as for the latter to be able to read user posts in Facebook and Twitter.

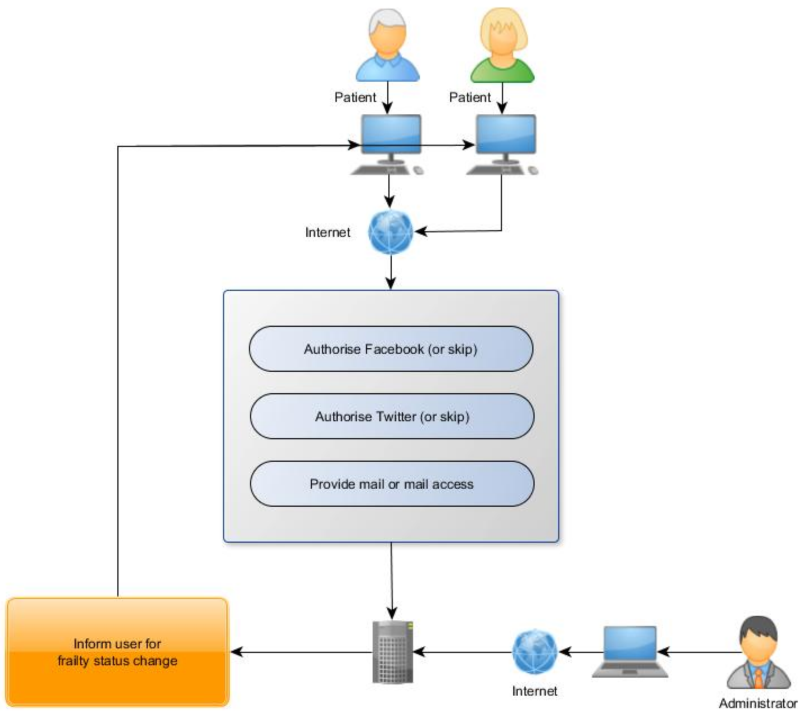
Patient
Patient
Internet
Authorise Facebook (or skip)
Authorise Twitter (or skip)
Provide mail or mail access
Inform user for frailty status change
Internet
Administrator

Figure 4: User flow

User (or patient for our case) must navigate through the following steps in order to complete his or her registration to the FrailSafe LingTester system.

- Step 1, user starts the process
    - User clicks “Sign up”

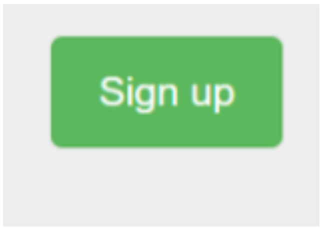

Figure 5: Step 1

- Step 2, Facebook authorisation (Facebook, 2017)
    - User clicks “Associate with Facebook” or “Skip”. The first option allows the user to authorise the FrailSafe App to access Facebook posts while the second option skips this step and redirects the user to the next step.

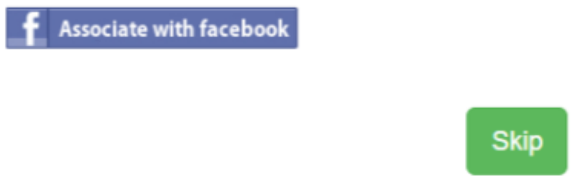

Figure 6: Step 2

- Step 3, Twitter authorisation (Twitter, 2017)
  - User clicks “Associate with Twitter” or “Skip”. The first option allows the user to authorise the FrailSafe App to access Twitter posts while the second option skips this step and redirects the user to the next step.

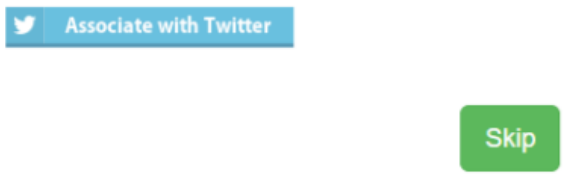

Figure 7: Step 3

- Step 4, email submission
  - User submits his email to receive notifications. The email, which can be different from the one connected to the Facebook account. Also, by saving the user’s email, the system can also retrieve email submissions through the mailbox and group all posts from the same user. For future reference, participant may also provide username and password (The Php Group, 2017)

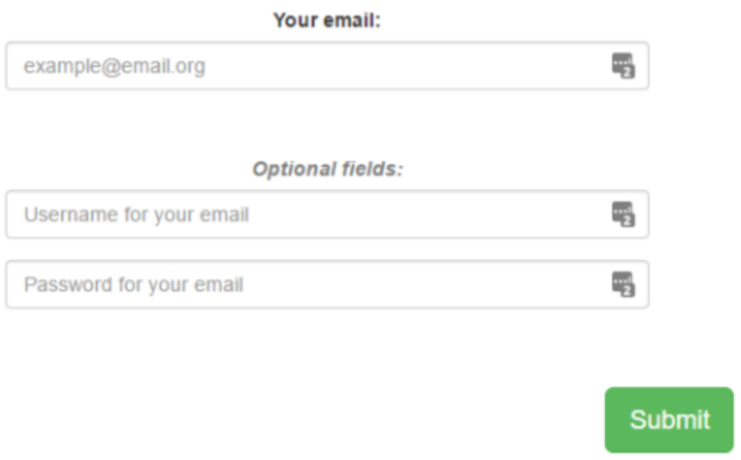

Figure 8: Step 4

- Step 5, final step
  - User concludes the process

3.7 Crawler flow

Every time the crawler executes it follows the steps as defined below to identify new Facebook submissions, new Twitter submissions and new mails received through the mailbox. The following figure (Figure 9), although simplified, tries as much as possible to show how the crawler behaves in each case.

At this point, we should stress that the overall behavior of the system and the crawler in specific is fully parametric and can be centrally defined through global constants, which affect the following actions:

- The user must have the same prediction multiple times, to assume safe prediction and avoid fluctuations.
- The user will be notified only after his/her frailty status changes from the previous time, and only in case the prediction is different than non-frail.
- The user will not be notified again for the same prediction if he/she has already been notified in the last few days, in order to avoid spamming the participant's mailbox. This way, we also overcome false positive in each mailbox spam filter

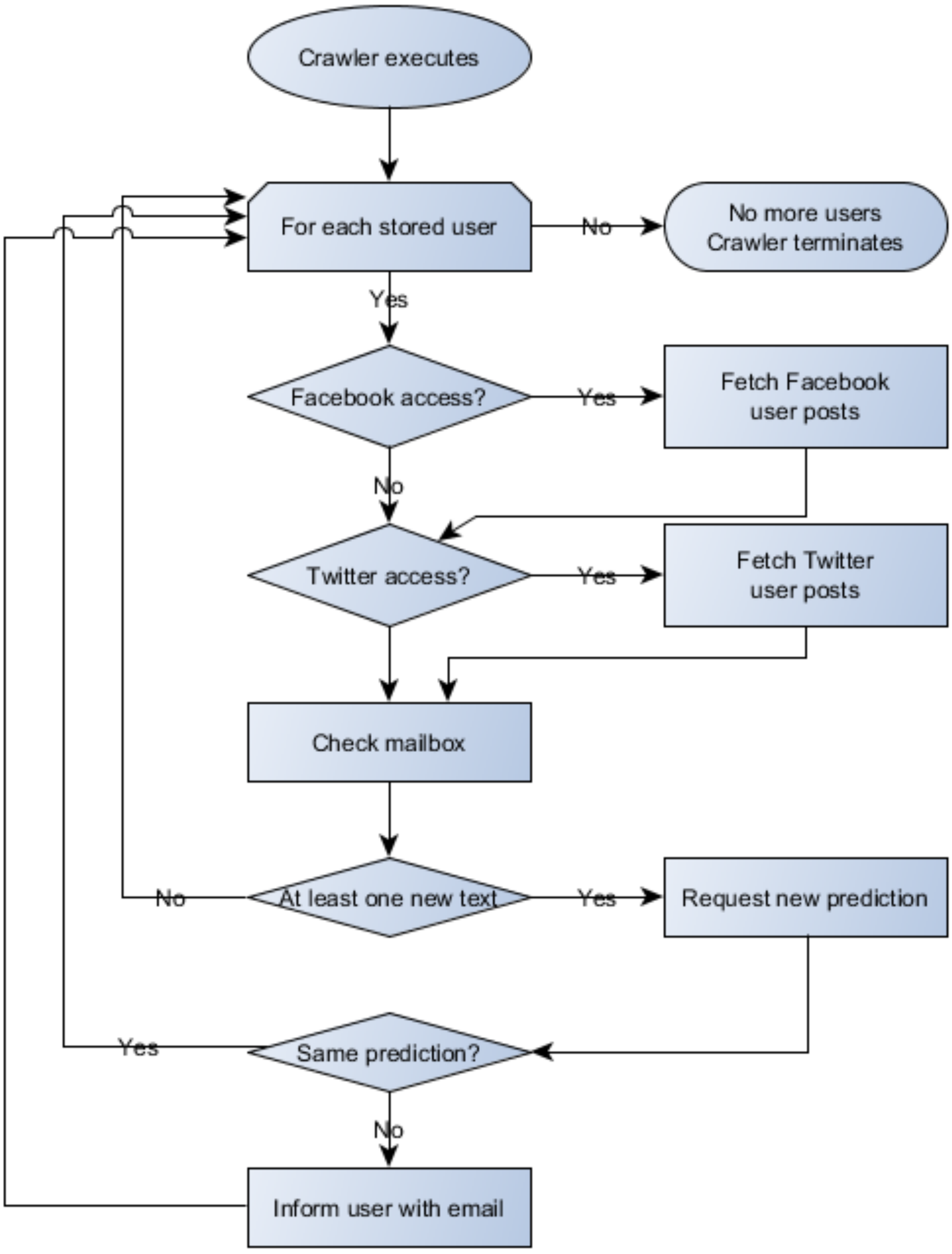

Figure 9: Crawler algorithmic procedure

4. Backend

The LingTester online mode uses the backend server to provide frailty predictions based on the user input that the frontend collects.

4.1 Architecture

The following image (Figure 10) shows how the backend works. The main service waits for an API call, and if one arrives (currently by the Crawler Application as shown in figure 2), it tries to make a prediction as fast as possible, in order to accept another one. API call can be made from ay host within the VPN network. Also, we decided that a watchdog service was needed to make sure that the prediction service will always be up and running *no matter what* may happen, in case of corrupted or buggy input.

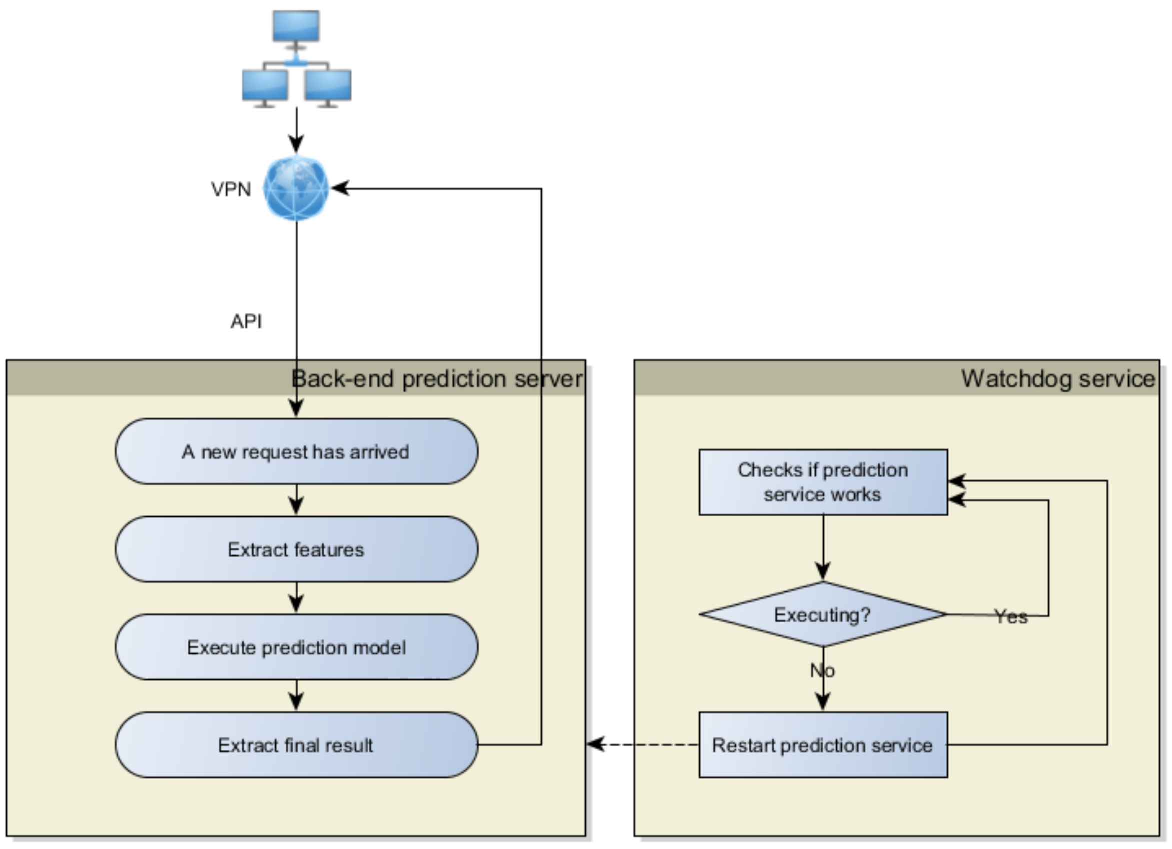
VPN
API
Back-end prediction server
A new request has arrived
Extract features
Execute prediction model
Extract final result
Watchdog service
Checks if prediction service works
Executing?
Yes
No
Restart prediction service

Figure 10: Backend architecture

4.2 Installation

The backend server has been installed on top of the Linux operating system. Linux was the obvious solution for such a service for numerous reasons. First of all, the initial feature extraction algorithm was created under Linux (as discussed in D4.10), so keeping the operating system the same was the natural way to go. As Python can be run under different operating systems, technical team decided that there was no reason to switch to a different operating system, as secondly and most importantly, Linux is stable in the long run for web services.

In order to install the aforementioned backend server from scratch the following steps should be reproduced. All commands must be run as root user, in order to overcome any permission hiccups. All commands, while self-explanatory for Linux administrators, are followed by a small description.

- `apt-get install default-jre`
    - Installs Java runtime environment

- `apt-get install python-pip`
    - Installs pip Python library, a helper library to install other Python libraries

- `pip install flask`
    - Install flask Python library, that runs a web service

- `pip install pyenchant`
    - Installs enchant Python library, used for spell checking

- `pip install httplib2`
    - Install httplib2 Python library, used for the feature extraction process, and the ability of this module to request third party URLs for example third party translation

- `pip install nltk`
    - Install Natural Language ToolKit python library, used for feature extraction and the tf-idf procedure

- `pip install pattern`
    - Install pattern Python library, used to extract sentiment score

- `pip install numpy`
- `pip install sklearn`
- `pip install scipy`
    - Install various libraries, for feature extraction and manipulation

- `apt-get install hunspell-el myspell-fr-fr hunspell-fr`

- Installs all necessary languages for the spell checker module

4.3 API

An Application Programming Interface has been implemented for easy backend access. There is no access control at this level, as this service is only accessible within the VPN(Feilner, 2006). It was decided to avoid adding access control, and let the service available to any request from within the VPN, for future use, by existing or new modules.

Expected Input

- Method
  - **POST**
- Format
  - **JSON**
- Arguments
  - oldText
    - *text from previous submissions for specific user*
  - newText
    - *current submission, on which prediction model must make a decision*

Expected Output

- Format
  - **JSON**
- Result
  - **Array**
- Contents of Array
  - oldTexttext
    - as given from input, for the caller to verify in case there are encoding errors
  - newText
    - text as given from input, for the caller to verify in case there are encoding errors
  - time
    - server time, for debugging purposes
  - prediction
    - result of the prediction model, *non-frail*, *pre-frail* or *frail*. In case something unexpected has happened, *na* is returned.
  - command
    - command executed internally through the operating system, for debugging purposes
  - raw_res
    - raw result from the model prediction executable, for debugging purposes

5. TEST RESULTS

5.1 Introduction

As this is the first prototype of the Online LingTester software, it is considered necessary to obtain the first test results from a series of debugging data. This chapter explains the exact process that was followed in order to obtain the test results. Clearly, these results are useful mainly for debugging reasons, so that the program flow can be validated mainly for it's correctness. Following the preliminary report, an attempt to run the software on real participant accounts will be made.

5.2 Participants

In order to run the test results for the prototype software, a number of artificial participant social media accounts were created. The participants, as has already been said, have no connection with the real FrailSafe program participants and this is because the software is still in a very early stage of development. All user accounts are kept private with no social interaction. At the end of the software testing period, all the accounts including their data will be removed.

The number of artificial participants is four. Each participant is referred by its user id. For each user, fake Facebook, Twitter and email account has been given. In order to be able to validate the obtained test results, the artificial participants have been assigned a frailty status that will match with the relating artificial data that will be created for them. Also, artificial age and gender have been assigned to them. The next table summarizes the artificial user accounts that were made.

USER ID	SERVICE	USERNAME	AGE	Gender	CLASS
1	facebook	user_1@lingtester.frailsafe-project.cloud	70	male	NONFRAIL
	twitter	user_1@lingtester.frailsafe-project.cloud			
	email	user_1@lingtester.frailsafe-project.cloud			
2	facebook	user_2@lingtester.frailsafe-project.cloud	75	female	PREFRAIL

	twitter	user_2@lingtester.frailsafe-project.cloud			
	email	user_2@lingtester.frailsafe-project.cloud			
3	facebook	user_3@lingtester.frailsafe-project.cloud	82	male	FRAIL
	twitter	user_3@lingtester.frailsafe-project.cloud			
	email	user_3@lingtester.frailsafe-project.cloud			
4	facebook	user_4@lingtester.frailsafe-project.cloud	74	female	N/A
	twitter	user_4@lingtester.frailsafe-project.cloud			
	email	user_4@lingtester.frailsafe-project.cloud			

Table 1: List of participants

For each one of the artificial users a text-posts profile will be created according to their assigned label. In more detail, random text phrases found in the FrailSafe database will be assigned to them, relating to their frailty status. To be clear, the used phrases have been recorded and classified during clinical examinations. The phrases simply give a description of an image or an event. All possible sensitive data that these phrases may include have already been removed by previous project tasks. For the last user found in the above table (user id: 4), no class has been assigned as this user will be profiled with random phrases unrelated to the FrailSafe database. This last user is impossible to be validated as its assigned class is unknown, thus it was created to generally demonstrate the user classification by the online LingTester tool using out-of-sample data.

5.3 Classification model

The prediction model(Michalski, 2013) used by the online LingTester tool has already been presented in deliverable D4.10. An almost identical but slightly tuned model is integrated to this tool. For this reason no details of its creation will be shown here. A table with the basic features it uses follows below.

Feature Names	Feature Description
Language	The crawled posts language
Sex	The gender of the participant
Number_of_words	The crawled text number of words
Text_entropy	The text entropy of the post

Crawled_text_ENG_sentiment	A sentiment score on the text
Prev_text_ENG_sentiment	Sentiment score of older text that was crawled
Crawled_text_misspeled	The misspellings score calculated on post text

Table 2: List of features of the prediction model

As regards the deployed algorithm, a pre-trained (D4.10) Decision Tree is used with its basic parameters as follows.

Parameter Name	**Parameter Value**
Binary splits	False
Confidence factor	0.25
MinNumObj	2
Reduced Error Pruning	False
Unpruned	True
Use Laplace	False

Table 3: Algorithmic parameters of the prediction model

5.4 Frailty tests & results

As described by the previous chapters, after the registration of a new user to the LingTester online tool fetches periodically the user's social activity. Furthermore, it analyzes the user's text posts and continuously monitors the user's mental frailty status. In order to test the system and its predictive abilities, the set of artificial users was utilized in the following process.

LingTester online - Testing process

---

1. For each artificial user:

a. Pick 3 phrases from the pool of the participants' phrases according to its assigned frailty status, based on the dataset obtained from D4.10, as provided by the clinical groups.

b. Create a private Facebook post with phrase 1

c. Create a private Twitter post with phrase 2

d. Send an email to the online LingTester tool with phrase 3

2. Expect and collect the frailty report from the LingTester tool

3. Repeat steps (1) and (2) for 4 times

4. Evaluate the assigned user's class with the reported user's class.

The size of evaluation data as well as the number of iterations that were selected, were limited by the amount of nonfrail data that were available. Therefore each artificial user was assigned twelve text-posts. Nevertheless, the number of posts is considered sufficient for an in sample test case for an already evaluated predictive model and an evaluation process that its main focus is to test the good working flow of the online software tool.

In the next table, a number of sample phrases with their respective class are shown for supervision.

Sample Phrase	**Class**
Το κορίτσι το οποίο βοηθά το αγόρι που είναι ανεβασμένο στο σκαμπό προκειμένου να φτάσει από το ντουλάπι της κουζίνας τρόφιμα.	NONFRAIL
μια κυρια πλενη πιατα και δυο παιδια κατεβαζου γλυκα Από το τουλαπη	PREFRAIL
Προφανώς η μητέρα η οποία πλένει και σκουπίζει τα πιάτα στο νεροχύτη της κουζίνας.	NONFRAIL
Ένα παιδί ανέβη γιά νά πάρη πισκοττα αλλά το σκαμνάκι έγιρε καί θά πέση καί ένα άλλο παιδί προσπαθή νά τον γλυτώση.	FRAIL
Κορίτσι αγόρι κάτι πέρνη καί θα γλιστρίσει	PREFRAIL
δυό παιδιά τον ένα πάνω σκαμπνί κ το κοριτιστάκι νά περιμενει νά του δώσει κάτι Μιά γυναίκα τα πλένει πιάτα	FRAIL
Παγκόσμια ημέρα νοσηλευτών σήμερα. Χρόνια πολλά σε όλους τους νοσηλευτές της χώρας.	N/A
Θα ήθελα να ευχαριστήσω όλους όσους παρευρέθηκαν και μας τίμησαν χτες βράδυ στην εκδήλωση που διοργάνωσε ο πολιτιστικός και αθλητικός σύλλογος μας.	N/A

Table 4: Test input

According to the algorithm executed, four predicted classes were obtained for each of the artificial users. For the first three users of the predicted classes only a minority of two classes was wrongly predicted giving an overall 83.33% accuracy(Huang, 2006). The last user, as explained can not be evaluated and was added to display a few out of sample predictions.

USER ID	PREDICTED CLASS				ACTUAL CLASS
	ITERATION 1	ITERATION 2	ITERATION 3	ITERATION 4	
1	NONFRAIL	NONFRAIL	NONFRAIL	NONFRAIL	NONFRAIL
2	PREFRAIL	FRAIL	PREFRAIL	NONFRAIL	PREFRAIL
3	FRAIL	FRAIL	FRAIL	FRAIL	FRAIL
4	NONFRAIL	PREFRAIL	NONFRAIL	NONFRAIL	N/A

Table 5: Prediction test results

5.5 Discussion of the results

As this is still a preliminary version of the online LingTester software tool, it was imperative to have a first testing and debugging procedure with artificial users. The results we obtained were generally in line with the FrailSafe dataset. The overall accuracy was 83.33% with a few of the predicted classes being wrong. This is an expected outcome as the integrated model is nearly the same with that of D4.10 but not exactly the same because of the restrictions and the reduction of information the whole flow of the online software tool introduces(e.g. some of the model feature values are not always available by the online users).

This testing process stands as a validation that the developed software is generally flawless and can be further utilized to obtain good results, as good as the integrated model can produce. It is surely a good starting point for further development and integration of more models and functionalities like suicidal tendencies detection and user text analysis reports. On the final version of the online LingTester tool a series of tests will be performed on real world case scenarios.

6. ETHICS AND SAFETY

Throughout the construction of the online Lingtester tool, legal issues were kept in mind so as to protect sensitive information. First of all, as described before, SSL is used between the participant and the frontend server, which ensures that the communication through internet providers is fully protected against unauthorised persons.

Furthermore, the user is fully informed and gives consent to provide any necessary access to third party social networks before signing up. In addition, each provided text is anonymised by stripping sensitive information before any other step. Moreover, communication between the frontend and backend servers is available strictly through a secure VPN.

The data obtained, is automatically filtered and all sensitive information is removed as discussed in chapter 3.4. No data is preserved prior to the anonymization process.

Finally, all emails sent to the predefined mail account are sent manually by each user, so consent is by default given for the full content, as it is the participants themselves that send the email towards the LingTester mailbox for further analysis.

7. REFERENCES

- J. Lee and B. Ware. Open source Web development with LAMP: using Linux, Apache, MySQL, Perl, and PHP. Addison-Wesley Professional, 2003.
- Nemeth, Evi, Garth Snyder, and Trent R. Hein. Linux administration handbook. Addison-Wesley Professional, 2006.
- I. Ristic, "Internet SSL Survey 2010," Talk at BlackHat 2010. Slides from https://media.blackhat.com/bh-us-10/presentations/Ristic/BlackHat-USA-2010-Ristic-Qualys-SSL-Survey-HTTP-Rating-Guide-slides.pdf, 2010, {online; last retrieved in May 2011}.
- MySQL, A. B. "MySQL." (2001).
- Add Facebook Login to Application or Website, https://developers.facebook.com/docs/facebook-login, Facebook, 2017
- Authentication & Authorization, https://dev.twitter.com/oauth/overview/authentication-by-api-family, Twitter, 2017
- PHP: IMAP Functions - Manual, http://php.net/manual/en/ref.imap.php, The Php Group, 2017
- Feilner, Markus. OpenVPN: Building and integrating virtual private networks. Packt Publishing Ltd, 2006.
- Michalski, Ryszard S., Jaime G. Carbonell, and Tom M. Mitchell, eds. Machine learning: An artificial intelligence approach. Springer Science & Business Media, 2013.
- Huang, Jin. Performance measures of machine learning. University of Western Ontario, 2006.

8. FILE STRUCTURE

These are the files that accompany this deliverable:

- Folder: frontend
    - Folder: crawler, *all needed files for the crawler to run based on cron*
        - File: index.php, *main crawler initialisation and loop*
        - File: lib.php, *library file with useful functions*
        - File: TwitterAPIExchange.php, *library file to retrieve twitters*
        - Folder: PHPMailer, *external PHPMailer library to send emails*
    - Folder: files, *secondary files for various uses*
    - Folder: images, *images for the User Interface (UI)*
    - Folder: catalog, *main files for the UI*
        - Folder: controller, *files for DB manipulation*
        - Folder: view, *files to construct UI based on PHP, JavaScript and stylesheets*
        - Folder: lib, *library files for third party modules and services*
    - File: config.php, *configuration file for site wide parameters*
    - File: frailsafe-online.sql, *initial SQL script to reconstruct the database*
    - File: index.php, *main file for UI*
- Folder: backend
    - File: frailsafe.model, *main model file of the prediction model in binary format*
    - File: offline_parser.py, *wrapper python file for feature extraction*
    - File: predictor-cli.jar, *source code of the demo predictor-cli.jar file*
    - File: runner.py, *web service wrapper of the offline_parser executor*
    - File: runner.sh, *watchdog wrapper of the main executable file runner.py*
    - File: SentiWordNet-1.txt, *sentiment analysis word list*
    - File: stemming.py, *text library file*

9. Annexes

9.1 SQL initial import script

```sql
SET names utf8;
SET time_zone = '+00:00';
SET foreign_key_checks = 0;
SET sql_mode = 'NO_AUTO_VALUE_ON_ZERO';
DROP TABLE IF EXISTS `events`;
CREATE TABLE `events` (
     `ea`          BIGINT(20) NOT NULL auto_increment,
     `iduser`  INT(11) NOT NULL,
     `when`    BIGINT(20) NOT NULL,
     `status`  VARCHAR(100) NOT NULL,
     `actions` VARCHAR(100) NOT NULL,
     `log`     TEXT NOT NULL,
     PRIMARY KEY (`ea`),
     KEY `iduser_when` (`iduser`, `when`),
     KEY `iduser_status_when` (`iduser`, `status`, `when`),
     CONSTRAINT `events_ibfk_1` FOREIGN KEY (`iduser`) REFERENCES `user` (
     `iduser`)
) engine=innodb DEFAULT charset=utf8;

DROP TABLE IF EXISTS `text_history`;
CREATE TABLE `text_history` (
     `aa`          INT(11) NOT NULL auto_increment,
     `iduser` INT(11) NOT NULL,
     `text`   TEXT NOT NULL,
     `utime`  BIGINT(20) NOT NULL,
     `lang`   VARCHAR(100) NOT NULL,
     PRIMARY KEY (`aa`),
     KEY `iduser_when` (`iduser`, `utime`),
     CONSTRAINT `text_history_ibfk_1` FOREIGN KEY (`iduser`) REFERENCES `user` (
     `iduser`)
) engine=innodb DEFAULT charset=utf8;

DROP TABLE IF EXISTS `user`;
CREATE TABLE `user` (
     `iduser`              INT(11) NOT NULL auto_increment,
     `userfacebookid`    TEXT,
     `usertwitterid`     TEXT,
     `userfacebookemail` TEXT,
     `usertwitteremail`  TEXT,
     `useremail`          TEXT CHARACTER SET utf16,
     `useremailusername` TEXT,
     `useremailpassword` TEXT,
     PRIMARY KEY (`iduser`)
) engine=innodb DEFAULT charset=utf8;
```

9.2 Frontend

9.2.a Requests Router

File: catalog/view/main.php

```html
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <meta name="viewport" content="width=device-width, initial-scale=1">

        <link href="image/favicon.ico" rel="shortcut icon" />
        <title>FrailSafe Online System</title>

            <!--css-->
            <link href="catalog/lib/bootstrap/css/bootstrap.min.css" rel="stylesheet" />
            <link href="catalog/view/stylesheet/jumbotron-narrow.css" rel="stylesheet" />
            <link href="catalog/view/stylesheet/index.css" rel="stylesheet" />

            <!--js-->
        <script type="text/javascript" src="catalog/lib/jquery/js/jquery-3.2.0.min.js"></script>
        <script type="text/javascript" src="catalog/lib/bootstrap/js/bootstrap.min.js"></script>
        <script type="text/javascript" src="catalog/view/js/index.js"></script>

    </head>
    <body>

        <div class="container">
            <?php
            include_once 'catalog/view/code/header.php';
            ?>

            <?php
            if(isset($_REQUEST['home']))
                include_once 'catalog/view/code/home.php';
            elseif(isset($_REQUEST['step1']))
                include_once 'catalog/view/code/step1.php';
            elseif(isset($_REQUEST['step2']))
                include_once 'catalog/view/code/step2.php';
            elseif(isset($_REQUEST['step3']))
                include_once 'catalog/view/code/step3.php';
            elseif(isset($_REQUEST['success']))
                include_once 'catalog/view/code/success.php';
            elseif(isset($_REQUEST['about']))
                include_once 'catalog/view/code/about.php';
            elseif(isset($_REQUEST['contact']))
                include_once 'catalog/view/code/contact.php';
            else
                include_once 'catalog/view/code/home.php';
            ?>


            <?php
            include_once 'catalog/view/code/footer.php';
            ?>

        </div>
        <!-- /container -->

    </body>
</html>
```

9.2.b Registration

File: catalog/lib/hybridauth/frailsafe/profile.php

```
<?php
    session_start();
    // config and whatnot
    $config = dirname(__FILE__) . '/../hybridauth/config.php';
    require_once( "../hybridauth/Hybrid/Auth.php" );

    $user_data = NULL;
    $returnUrl = urldecode($_GET['returnurl']);
    //echo $_GET['returnurl'];
    //return;

    // try to get the user profile from an authenticated provider
    try{
        $hybridauth = new Hybrid_Auth( $config );

        // selected provider name
        $provider = @ trim( strip_tags( $_GET["provider"] ) );

        // check if the user is currently connected to the selected provider
        if( ! $hybridauth->isConnectedWith( $provider ) ){
            // redirect him back to login page
            header( "Location: login.php?error=Your are not connected to $provider or your session has expired" );
        }

        // call back the requested provider adapter instance (no need to use authenticate() as we already did on login page)
        $adapter = $hybridauth->getAdapter( $provider );

        // grab the user profile
        $user_data = $adapter->getUserProfile();
        //session_destroy();
        //session_start();
        include_once('../../../../config.php');
        include_once('../../../../catalog/controller/dbOperator.php');
        $dbOp = new dbOperator($dbHost,$dbUsername,$dbPassword,$dbName);

        $user = $dbOp->query("Select * from user where user".$provider."Email='".$user_data->email."' LIMIT 1");
        //print_r($user);
        if(!$user->num_rows){
            echo 'not registered';
            $id = register($dbOp,$user_data,$provider);
        }else
            //$id = $user[0]['idUser'];
            $id = $user->row['idUser'];

        if(strpos($returnUrl, "step1"))
            $returnUrl = str_replace("step1", "step2", $returnUrl);
        elseif(strpos($returnUrl, "step2"))
            $returnUrl = str_replace("step2", "step3", $returnUrl);

        if(!isset($_SESSION['idUser'])){
            $_SESSION['idUser'] = $id;

            //$urlParameter ='';
            //header("Location: ".((strpos($returnUrl,'?'))?urldecode($returnUrl).'&'.$urlParameter:urldecode($returnUrl).'?'.$urlParameter) );
            header("Location: ".urldecode($returnUrl));
        }else
            header("Location: ".urldecode($returnUrl));

    }
    catch( Exception $e ){
        // In case we have errors 6 or 7, then we have to use Hybrid_Provider_Adapter::logout() to
        // let hybridauth forget all about the user so we can try to authenticate again.

        // Display the recived error,
        // to know more please refer to Exceptions handling section on the userguide
        switch( $e->getCode() ){
            case 0 : echo "Unspecified error."; break;
```

```php
                case 1 : echo "Hybriauth configuration error."; break;
                case 2 : echo "Provider not properly configured."; break;
                case 3 : echo "Unknown or disabled provider."; break;
                case 4 : echo "Missing provider application credentials."; break;
                case 5 : echo "Authentication failed. "
                                    . "The user has canceled the authentication or the provider refused the connection.";
                case 6 : echo "User profile request failed. Most likely the user is not connected "
                                    . "to the provider and he should to authenticate again.";
                        $adapter->logout();
                        break;
                case 7 : echo "User not connected to the provider.";
                        $adapter->logout();
                        break;
        }

        echo "<br /><br /><b>Original error message:</b> " . $e->getMessage();

        echo "<hr /><h3>Trace</h3> <pre>" . $e->getTraceAsString() . "</pre>";
    }


    function register($dbOperator,$userData,$provider =''){
        $dbOp =$dbOperator;
        $user_data = $userData;

        if(!isset($_SESSION['idUser'])){
            if($dbOp->query("INSERT INTO `user`(`idUser`, `user".$provider."Id`, `user".$provider."Email`) VALUES ('','".$user_data->identifier."','".$user_data->email."')"))
                $id = $dbOp->getLastId();

        }elseif(!empty($_SESSION['idUser'])){
            $dbOp->query("UPDATE user SET `user".$provider."Id`='".$user_data->identifier."', `user".$provider."Email`='".$user_data->email."' WHERE idUser=".$_SESSION['idUser']);
            return $_SESSION['idUser'];
        }
        return $id;
    }
?>
```

9.3 Backend

9.3.a Crawler main loop

File: crawler/index.php

```php
<?php

include_once '../config.php';
$auth_config = include_once '../catalog/lib/hybridauth/hybridauth/config.php';
```

```php
include_once '../catalog/controller/dbOperator.php';
include_once dirname(__FILE__) . '/TwitterAPIExchange.php';
include_once dirname(__FILE__) . '/lib.php';
include_once dirname(__FILE__) . '/PHPMailer/PHPMailerAutoload.php';

// Within this window, text will be considered present
define('CURRENT_TEXT_WINDOW', 60 * 60 * 24);

// Minimm same predictions in a raw to assume final
define('SAME_PREDICTIONS_IN_A_RAW', 5);

// If so much time has passed, let's send another email to the patient
define('NOTIFY_AGAIN_AFTER_DAYS', 30);
define('NOTIFY_WITH_TAGS', array('frail', 'prefrail'));

error_reporting(E_ALL);
ini_set('display_errors', 1);

//Create a new PHPMailer instance
$mail = new PHPMailer;

//Tell PHPMailer to use SMTP
$mail->isSMTP();

//Enable SMTP debugging
// 0 = off (for production use)
// 1 = client messages
// 2 = client and server messages
// $mail->SMTPDebug = 2;

//Ask for HTML-friendly debug output
$mail->Debugoutput = 'html';

//Set the hostname of the mail server
$mail->Host = 'smtp.gmail.com';
// use
// $mail->Host = gethostbyname('smtp.gmail.com');
// if your network does not support SMTP over IPv6

//Set the SMTP port number - 587 for authenticated TLS, a.k.a. RFC4409 SMTP submission
$mail->Port = 587;

//Set the encryption system to use - ssl (deprecated) or tls
$mail->SMTPSecure = 'tls';

//Whether to use SMTP authentication
$mail->SMTPAuth = true;

//Username to use for SMTP authentication - use full email address for gmail
$mail->Username = $auth_config['providers']['Google']['keys']['username'];

//Password to use for SMTP authentication
$mail->Password = $auth_config['providers']['Google']['keys']['password'];

//Set who the message is to be sent from
$mail->setFrom($auth_config['providers']['Google']['keys']['username'], 'FrailSafe Lingtester');


$dbOp = new dbOperator($dbHost, $dbUsername, $dbPassword, $dbName);

// Go through all users and fetch new texts
// .. and for new texts, get a new prediction
$users_with_new_texts = array();

// Facebook
// Create an access token using the APP ID and APP Secret.
$accessToken = $auth_config['providers']['Facebook']['keys']['id'] . '|' . $auth_config['providers']['Facebook']['keys']['secret'];

// Twitter
$twitterURL = 'https://api.twitter.com/1.1/statuses/user_timeline.json';
$twitterSettings = array(
  'oauth_access_token' => $auth_config['providers']['Twitter']['keys']['access_token'],
  'oauth_access_token_secret' => $auth_config['providers']['Twitter']['keys']['access_token_secret'],
  'consumer_key' => $auth_config['providers']['Twitter']['keys']['key'],
  'consumer_secret' => $auth_config['providers']['Twitter']['keys']['secret'],
);
$twitter = new TwitterAPIExchange($twitterSettings);

```

```php
$all_users = $dbOp->query('Select * from user;');
print '<pre>';
// print_r($all_users);

foreach ($all_users->rows as $user) {

  // Facebook
  if ($user['userFacebookId'] && $user['userFacebookId'] != '') {
    $userFacebookId = $user['userFacebookId'];

    // Tie it all together to construct the URL
    $url = sprintf('https://graph.facebook.com/%s/posts?access_token=%s', $userFacebookId, $accessToken);

    // Make the API call
    $result = file_get_contents($url);

    if ($result) {
      // Decode the JSON result.
      $decoded = json_decode($result, true);

      if ($decoded) {
        foreach ($decoded['data'] as $value) {
          // "Useless" posts
          if (!isset($value['message']))
            continue;

          $utime = strtotime($value['created_time']);
          $existing_post = $dbOp->query('SELECT * FROM text_history WHERE idUser = "' . $user['idUser'] . '" AND utime = "' . $utime . '" LIMIT 1;');

          if ($existing_post && $existing_post->num_rows > 0)
            continue;

          // Keep it handy
          $users_with_new_texts[ $user['idUser'] ] = 1;

          $text = $value['message'];
          $text = $dbOp->escape($text);
          $dbOp->query(sprintf('INSERT INTO text_history (`idUser`, `text`, `utime`, `lang`) VALUES (%d, "%s", %d, "");', $user['idUser'], $text, $utime));
        } // foreach ($decoded['data'] as $value)
      } // if ($decoded)
    } // if ($result)
  } // if ($user['userFacebookId'] && $user['userFacebookId'] != '')

  // Twitter
  if ($user['userTwitterId'] && $user['userTwitterId'] != '') {
    $userTwitterId = $user['userTwitterId'];

    $ret = $twitter->setGetfield('?user_id=' . $userTwitterId)
        ->buildOauth($twitterURL, 'GET')
        ->performRequest();

    if ($ret && is_array($ret)) {
      foreach ($ret as $tweet) {
        $utime = strtotime($tweet->created_at);
        $lang = $tweet->lang;
        $text = $tweet->text;
        $existing_post = $dbOp->query('SELECT * FROM text_history WHERE idUser = "' . $user['idUser'] . '" AND utime = "' . $utime . '" LIMIT 1;');

        if ($existing_post && $existing_post->num_rows > 0)
          continue;

        // Keep it handy
        $users_with_new_texts[ $user['idUser'] ] = 1;

        $text = $dbOp->escape($text);
        $dbOp->query(sprintf('INSERT INTO text_history (`idUser`, `text`, `utime`, `lang`) VALUES (%d, "%s", %d, "%s");', $user['idUser'], $text, $utime, $lang));
      } // foreach ($ret as $tweet)
    } // if ($ret && is_array($ret))
  } // if ($user['userTwitterId'] && $user['userTwitterId'] != '')
} // foreach ($all_users as $user)

$inbox = imap_open(
    $auth_config['providers']['Google']['connection'],
    $auth_config['providers']['Google']['keys']['username'],
    $auth_config['providers']['Google']['keys']['password']
);

```

```php
if ($inbox) {
  $emails = imap_search($inbox, 'UNSEEN');
  // $emails = imap_search($inbox, 'ALL');

  // if emails are returned, cycle through each..
  if ($emails) {

    // for every email...
    foreach($emails as $email_number) {
      // Get information specific to this email
      $overview = imap_fetch_overview($inbox, $email_number, 0);
      $message = imap_fetchbody($inbox, $email_number, '1');

      $from = $overview[0]->from;
      $utime = strtotime($overview[0]->date);

      // Iparxei kaneis stin vasi mas me afto to email?
      $matches = array();
      if (preg_match('/(.*?)\\<(.*?)\\>/', $from, $matches))
        $from = $matches['2'];

      $from = $dbOp->escape($from);
      $select = sprintf('SELECT * from user WHERE userFacebookEmail = "%s" OR userTwitterEmail = "%s" OR userEmail = "%s" LIMIT 1;', $from, $from, $from);
      $known_user = $dbOp->query($select);

      // No known user,
      if (!$known_user || $known_user->num_rows <= 0) {
        // TODO
        // .. send a reply that user must visit the page
        // .. https://lingtester.frailsafe-project.cloud/
        $mail->ClearAddresses();
        $mail->addAddress($from);
        $mail->Subject = 'FrailSafe: Lingtester prediction';
        $mail->Body = 'You are not in out system. Please visit https://lingtester.frailsafe-project.cloud/ and follow the steps.';
        $mail->send();
        continue;
      }

      $user = $known_user->row;

      // This is reduntant, as mails from POP are always returned once
      $existing_post = $dbOp->query('SELECT * FROM text_history WHERE idUser = "' . $user['idUser'] . '" AND utime = "' . $utime . '" LIMIT 1;');
      if ($existing_post && $existing_post->num_rows > 0)
        continue;

      // Keep it handy
      $users_with_new_texts[ $user['idUser'] ] = 1;

      $message = $dbOp->escape($message);
      $dbOp->query(sprintf('INSERT INTO text_history (`idUser`, `text`, `utime`, `lang`) VALUES (%d, "%s", %d, "%s");', $user['idUser'], $message, $utime, ''));
    } // foreach($emails as $email_number)
  } // if ($emails)

  // close the connection
  imap_close($inbox);
} // if ($inbox)

if (count($users_with_new_texts) <= 0)
  return;

foreach ($users_with_new_texts as $key => $value) {
  // This user has some new text
  // .. take all availiable and fetch a prediction
  $user_texts = $dbOp->query('SELECT * FROM text_history WHERE idUser = "' . $key . '";');
  $new_texts = '';
  $old_texts = '';
  // strip_tags is used to remove html tags
  // .. which can be found in posts and html mails
  foreach ($user_texts->rows as $value) {
    if ($value['utime'] >= time() - CURRENT_TEXT_WINDOW)
      $new_texts .= ' ' . strip_tags($value['text']);
    else
      $old_texts .= ' ' . strip_tags($value['text']);
  }

  // All texts considered, get a prediction
  // .. save it in the database
  $ret = getPrediction($old_texts, $new_texts, $backend);
  // Backend issue?
  if (!$ret)
    continue;

  $data = json_decode($ret);
  // Backend issue?
  if (!$data || !isset($data->prediction))
    continue;

  // Model issue?
  if (!in_array($data->prediction, array('frail', 'prefrail', 'nonfrail')))
    continue;

  $status = $data->prediction;
  $utime = time();
  $log = $dbOp->escape(serialize($data));
  $sql = sprintf('INSERT INTO events (`idUser`, `when`, `status`, `log`) VALUES (%d, %d, "%s", "%s");', $key, $utime, $status, $log);
  $dbOp->query($sql);

  // In order to send a message
  // .. we must have a different prediction than before
  // .. we must not have sent already an email to avoid spamming
  // .. where a new prediction means SAME_PREDICTIONS_IN_A_RAW all the time
  $new_prediction = '';
  $how_many_times = 0;
  $sql = sprintf('SELECT `ea`, `when`, `actions`, `status` FROM `events` WHERE `idUser` = %d ORDER BY `when` DESC;', $key);
  $prediction_history = $dbOp->query($sql);
  foreach ($prediction_history->rows as $value) {
    if ($how_many_times == 0 || $new_prediction == $value['status']) {
      $how_many_times += 1;
      $new_prediction = $value['status'];
    } // if ($how_many_times == 0 || $new_prediction == $value['status'])
  } // foreach ($prediction_history->rows as $value)

  // This doesn't qualify for notification yet
  if ($how_many_times < SAME_PREDICTIONS_IN_A_RAW || $new_prediction == '')
    continue;

  // No need to notify
  if (!in_array($new_prediction, NOTIFY_WITH_TAGS))
    continue;

  // We must notify patient
  // .. but have we already yet?
  $last_email_sent = Null;
  $with_prediction = '';
  foreach ($prediction_history->rows as $value) {
    if (stripos($value['actions'], 'send-email') !== False) {
      $last_email_sent = $value['when'];
      $with_prediction = $value['status'];

      // We are in descending order
      // .. nothing to check more
      break;
    }
  } // foreach ($prediction_history->rows as $value)

  // Patient has already been notified within a considerable amount of time
  if ($with_prediction == $new_prediction && time() - intval($last_email_sent) < NOTIFY_AGAIN_AFTER_DAYS * 60 * 60 * 24)
    continue;

  // At this point we must definitely send an email
  $user_row = $dbOp->query('SELECT * FROM `user` WHERE idUser = "' . $key . '";');

  $mail->ClearAddresses();
  if (isset($user_row->row['userFacebookEmail']))
    $mail->addAddress($user_row->row['userFacebookEmail']);
  if (isset($user_row->row['userTwitterEmail']))
    $mail->addAddress($user_row->row['userTwitterEmail']);
  if (isset($user_row->row['userEmail']))
    $mail->addAddress($user_row->row['userEmail']);

  $mail->Subject = 'FrailSafe: Lingtester prediction';
```

```php
    $mail->Body = 'Your current prediction is ' . $new_prediction;
    if($mail->send()) {
      // Update database to avoid spamming in the future
      $actions = explode(',', $prediction_history->row['actions']);
      $actions[] = 'send-email';
      $ea = $prediction_history->row['ea'];
      $sql = sprintf('UPDATE `events` SET `actions` = "%s" WHERE `ea` = %d;', implode(',', $actions), $ea);
      $update_system = $dbOp->query($sql);
    }

    print_r($prediction_history);
} // foreach ($users_with_new_texts as $key => $value)
```

9.3.b Text Parser

```python
# -*- coding: utf-8 -*-

from flask import Flask
from flask import request

import os
import offline_parser
import json
import time

app = Flask(__name__)

def identifyLang(Text):
    if Text.find(u'α') > 0:
      return 'greek'

    return 'english'

def create_test_arff(oldText, newText, relation = 'frailsafe_111'):
    """Create arff for WEKA with all features availiable
   """
    out = []
    out.append('@RELATION %s' % relation)
    out.append('')

    basic_tags = []
    basic_tags.append('transcript')
    basic_tags.append('tag')
    basic_tags.append('sex')

    for tag in basic_tags:
        valid = offline_parser.verify_tags['-' + tag]
        if tag == 'tag':
            tag = 'class'
            # valid = ('nonfrail', 'prefrail', 'frail')
            valid = ('prefrail', 'frail')
        out.append('@ATTRIBUTE %s {%s}' % (tag, ','.join(valid)))

    out.append('@ATTRIBUTE %s %s' % (offline_parser.get_feature_word_count('', 'title'), offline_parser.get_feature_word_count('', 'type')))
    out.append('@ATTRIBUTE %s %s' % (offline_parser.get_feature_text_shannon_entropy('', 'title'), offline_parser.get_feature_text_shannon_entropy('', 'type')))

    for tag in ['-desc_event_ENG', '-prev_text_ENG']:
        out.append('@ATTRIBUTE %s %s' % (tag.lstrip('-') + '_sentiment', 'real'))

    for tag in ['-desc_image', '-desc_event']:
        out.append('@ATTRIBUTE %s %s' % (tag.lstrip('-') + '_misspelled', 'real'))

    texts = []
    text_POS = []

    out.append('')
    out.append('@DATA')
    out.append('')

    filename = 'in.arff'
    f = open(filename, 'w')
    f.write("\n".join(out).encode('utf8'))
    f.write("\n")

    clang = identifyLang(oldText + newText)

    # To absorb all Greek variations
    if clang.startswith('greek'):
        clang = 'greek'

    row = []
    # transcript?
    row.append('no')
    # sex?
    row.append('?')
    # tag?
    row.append('?')

    row.append(str(offline_parser.get_feature_word_count(oldText + newText, lang = clang)))
    row.append(str(offline_parser.get_feature_text_shannon_entropy(oldText + newText, lang = clang)))

    # Sentiment score is based in the english translation
    for tag in ['-desc_event_ENG', '-prev_text_ENG']:
        text = ''
        if tag.find('event') > 0:
          text = newText
        elif tag.find('prev') > 0:
          text = oldText

        # Get sentiment score
        eng_text = offline_parser.get_translated_data(text, clang)
        ret = offline_parser.get_feature_sentiment_score(eng_text)

        row.append(str(ret))

    for tag in ['-desc_image', '-desc_event']:
        text = ''
        if tag.find('event') > 0:
          text = newText
        elif tag.find('prev') > 0:
          text = oldText

        ret = offline_parser.get_feature_mispelling_score(text, lang = clang)
        row.append(str(ret))

    f.write(','.join(row).encode('utf8'))
    f.write("\n")

    f.close()

@app.route("/", methods=['POST'])
def predict():
    oldText = request.form['oldText']
    newText = request.form['newText']

    create_test_arff(oldText, newText)

    temp_file_out = "%d.txt" % time.time()
    command = "java -jar predictor-cli.jar > %s 2>&1" % temp_file_out
    os.system(command)
    res = ''
    f = open(temp_file_out)
    res = "\n".join(f.readlines())
    f.close()

    os.unlink(temp_file_out)

    ret = {}
    ret['oldText'] = oldText
    ret['newText'] = newText
    ret['time'] = temp_file_out
    ret['prediction'] = 'na'
    ret['command'] = command
    ret['raw_res'] = res

    if res.find('Prediction for instance: 0 is:') >= 0:
      ret['prediction'] = res.replace('Prediction for instance: 0 is:', '').strip().lower()

    return json.dumps(ret)

if __name__ == "__main__":
    app.run(host = '0.0.0.0', )
```

9.4 Predictor

File: predictor/predictor-cli.java

```java
package predictor;

import weka.classifiers.Classifier;
import weka.core.Instances;
import weka.core.converters.ConverterUtils.DataSource;

public class PredictorCLI {

    public static void main(String[] args) {

        Classifier cls;
        try {
            //load model
            cls = (Classifier) weka.core.SerializationHelper.read("frailsafe.model");


            DataSource source;
            try {
                //load test data
                source = new DataSource("in.arff");
                Instances data = source.getDataSet();
                if (data.classIndex() == -1)
                   data.setClassIndex(1); //class attribute is the second attribute

                //predict & print
                for(int i=0; i<data.numInstances();i++){
                    double value=cls.classifyInstance(data.instance(i));
                    String prediction=data.classAttribute().value((int)value);
                    System.out.println("Prediction for instance: "+i+" is: "+prediction);
                }

            } catch (Exception e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }

}
```