



**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: D4.9**

### **LingTester Test Results – Active (on-line) mode**

**Due date of deliverable:** M24 (30<sup>th</sup> June 2017)

**Actual submission date:** 31<sup>st</sup> December 2017

**Version:** 1

**Date:** 31<sup>st</sup> December, 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 (vers a)
0.2	01/05/2017	draft	C. Tsimpouris (UoP), N. Fazakis (UoP), C. Makrhs (UoP)	First draft deliverable report (vers a)
0.3	12/06/2017	draft	C. Tsimpouris (UoP), N. Fazakis (UoP), C. Makrhs (UoP)	Updated deliverable report sent for internal review (vers a)
0.4	22/06/2017	draft	C. Tsimpouris (UoP), N. Fazakis (UoP),	Second draft deliverable report (vers a)
0.5	23/06/2017	draft	I. Kalamaras (CERTH), A. Vasilakis (CERTH)	Revision of the document (vers a)
0.6	30/06/2017	final	C. Tsimpouris (UoP), N. Fazakis (UoP), C. Makrhs (UoP)	Deliverable 4.8 finalised taking into account internal review's comments (vers a)
0.7	09/11/2017	draft	C. Tsimpouris (UoP),	Revision of the document (vers b)
0.8	27/11/2017	draft	N. Fazakis (UoP)	Revision of the document (vers b)
1.0	10/12/2017	draft	C. Tsimpouris (UoP), N. Fazakis (UoP), C. Makrhs (UoP)	Deliverable 4.9 finalised taking into account internal review's comments (vers b)

## EXECUTIVE SUMMARY

LingTester is the FrailSafe language analysis tool that aims to process the user's typed text and detect abnormal behaviour. The prototype software tool is in final stage and it is able to continuously monitor participants' activity in the supported social and communication platforms in order to detect possible suicidal manifestations and also 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 **D4.11**. 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

<b>Contract Number:</b>	H2020-PHC–690140	<b>Acronym:</b>	FRAILSAFE
<b>Full title</b>	Sensing and predictive treatment of frailty and associated co-morbidities using advanced personalized models and advanced interventions		
<b>Project URL</b>	<a href="http://frailsafe-project.eu/">http://frailsafe-project.eu/</a>		
<b>EU Project officer</b>	Mr. Ramón Sanmartín Sola		

<b>Deliverable number:</b>	4.9	<b>Title:</b>	LingTester Test Results – Active (online) mode
<b>Work package number:</b>	4	<b>Title:</b>	Data Management and Analytics

Date of delivery	Contractual	31/12/2017 (M24)	Actual	TBC
Status	Draft <input type="checkbox"/>		Final <input checked="" type="checkbox"/>	
Nature	Report <input checked="" type="checkbox"/> Demonstrator <input type="checkbox"/> Other <input type="checkbox"/>			
Dissemination Level	Public <input type="checkbox"/> Consortium <input checked="" type="checkbox"/>			
Abstract (for dissemination)	This deliverable reports on the choices made in the design of the online LingTester system, subsystems, 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			

<b>Contributing authors (beneficiaries)</b>	Tsimpouris Charalampos(UoP) Fazakis Nikos (UoP) Makrhis Xrhistos (UoP) Megaloioikonomou Vasileios (UoP)		
<b>Responsible author(s)</b>	Makrhis Xrhistos		<b>Email</b> makri@ceid.upatras.gr
	<b>Beneficiary</b>	UoP	<b>Phone</b> +30 2610 996968

# TABLE OF CONTENTS

CHANGE HISTORY	2
EXECUTIVE SUMMARY	3
DOCUMENT INFORMATION	3
TABLE OF CONTENTS	5
LIST OF FIGURES	7
LIST OF TABLES	7
LIST OF ANNEXES	7
<b>1. Introduction</b>	<b>8</b>
<b>2. Overall Architecture</b>	<b>8</b>
<b>3. Frontend</b>	<b>9</b>
3.1 Architecture	9
3.2 Installation	10
3.3 Communication privacy	11
3.4 Anonymization of data	12
3.5 MySQL Database schema	12
3.6 NoSQL SaaS	13
3.7 eCRF user authentication	14
3.8 Credentials Encryption	15
3.9 User-flow	15
3.10 Crawler flow	18
<b>4. Backend</b>	<b>20</b>
4.1 Architecture	20
4.2 Installation	21
4.3 API	22
<b>5. Test Results</b>	<b>23</b>
5.1 Introduction	23
5.2 Participants	23
5.3 Classification model	25
5.3 Suicidal model tests & results	28
5.5 Frailty tests & results	29
5.6 Discussion of the results	32
<b>6. Ethics and Safety</b>	<b>33</b>
<b>7. References</b>	<b>33</b>
<b>8. File structure</b>	<b>35</b>
<b>9. Annexes</b>	<b>36</b>
9.1 SQL initial import script	36
9.2 Frontend	36

9.2.a Requests Router	36
9.2.b Registration	37
9.3 Backend	40
9.3.a Crawler main loop	40
9.3.b Text Parser	44
9.4 Predictor	48
9.5 FrailSafe demo video	49

## LIST OF FIGURES & IMAGES

Figure 1: Online mode of LingTester	8
Figure 2: Frontend server, internal architecture	9
Figure 3: MySQL EER diagram.	12
Figure 4: User flow	16
Figure 5: Step 1	16
Figure 6: Step 2	17
Figure 7: Step 3	17
Figure 8: Step 4	17
Figure 9: Step 5	18
Figure 10: Crawler algorithmic procedure	19
Figure 11: Backend architecture	20
Image 1. Command line results	28

## LIST OF TABLES

Table 1: List of participants	24
Table 2: List of features of the prediction model	25
Table 3: Algorithmic parameters of the prediction model	27
Table 4: Test input	29
Table 5: Prediction test results	32

## LIST OF ANNEXES

9.1 SQL initial import script	36
9.2.a Requests Router	36
9.2.b Registration	37
9.3.a Crawler main loop	40
9.3.b Text Parser	44
9.4 Predictor	48
9.5 FrailSafe demo video	49

# 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 updated 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](#)).

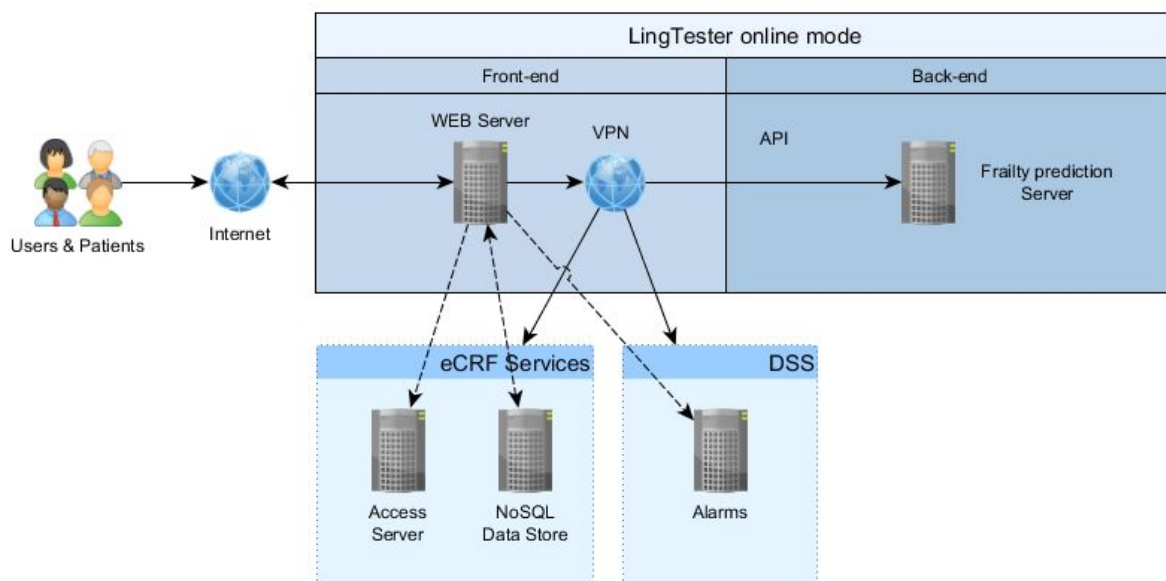


Figure 1: Online mode of LingTester



### 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 well-known as *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. As it will be shown below, MySQL is not utilised as all data is stored outside the server, through a Software as a Service (SaaS) approach.

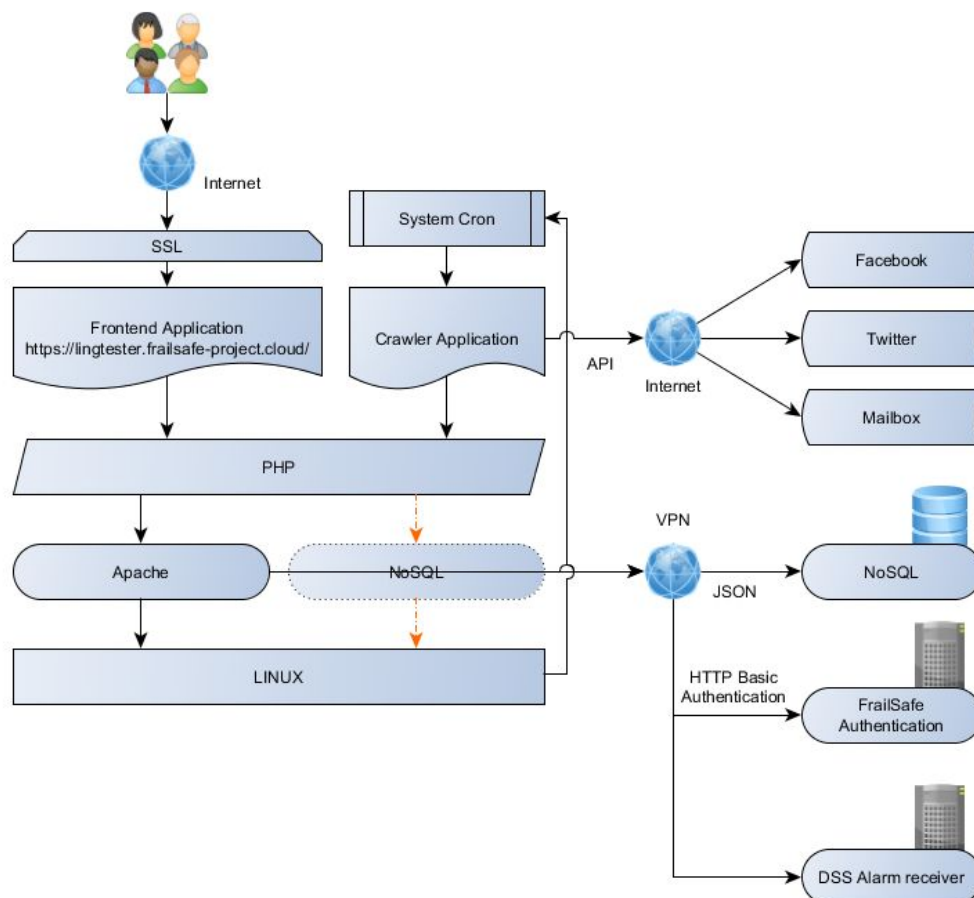


Figure 2: Frontend server, internal architecture

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

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
- `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/`
- Update all details in the file `config.php` so as to provide access as needed

---

<sup>1</sup> 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 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.

- 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`**
- NoSQL data storage, SaaS
  - Currently set as
    - **`http://data-analysis-external.frailsafe-project.cloud:5050`**
- DSS alert receiver
  - Currenty set as
    - **`http://data-analysis-external.frailsafe-project.cloud:5050`**
- eCRF authorization server:
  - Currently set as
    - **`https://api.frailsafe-project.cloud/auth/v1/login`**

### 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<sup>2</sup> (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 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<sup>3</sup>, 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,

<sup>2</sup> [https://en.wikipedia.org/wiki/Transport\\_Layer\\_Security](https://en.wikipedia.org/wiki/Transport_Layer_Security)

<sup>3</sup> <https://letsencrypt.org/>

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) was constructed to support all needed actions, store anonymised data and keep track of history events. The following image shows the EER diagram<sup>4</sup> as long as MySQL was used. *However, the following details are provided for consistency reasons in order for the reader to understand how we transitioned towards the NoSQL database.*

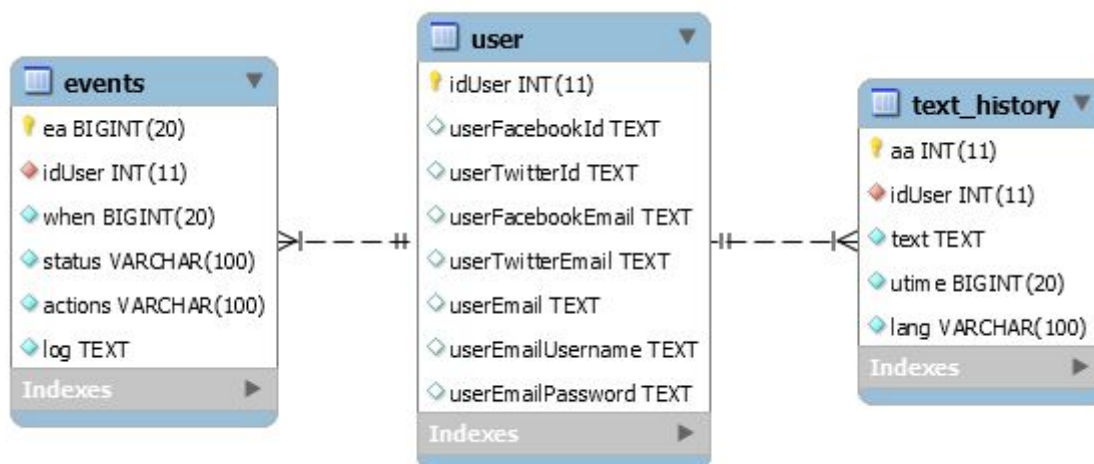


Figure 3. MySQL EER diagram.

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

<sup>4</sup> 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.

- *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<sup>5</sup>, 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 field 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 NoSQL SaaS

As mentioned in the previous chapter, MySQL was replaced by eCRF NoSQL data storage. However, and as state of the art in DB management dictates, a SaaS approach was selected for easy scalability and upgradeability that are not application-related while resilience against hardware failure is not a concern within the hardware of this application.

---

<sup>5</sup> [https://en.wikipedia.org/wiki/Unix\\_time](https://en.wikipedia.org/wiki/Unix_time)

The following API Calls were implemented, along with their description, and each call is towards the NoSQL server, according to **D4.16**. All communication is wrapped in JavaScript Object Notation (JSON<sup>6</sup>) format:

- *function getAllIDs()*
  - `/social/fetch_ids`: Returns all available participant IDs that system knows about already
- *function removeID(\$id)*
  - `/social/delete_all/id`: Deletes all written text for a specific participant ID from the server. This call is used mostly for debugging and cleaning but it can also be used if participant wants to be removed from the database
- *function getInfoById(\$id)*
  - `/social/fetch_info/id`: Retrieves all data stored within the NoSQL server, already stored through the `save_info` function as listed below. This include all data provided by the participant through the 5 steps, as shown in a previous chapter 3.7, for example Facebook and Twitter access token, extra email and other useful information for future process.
- *function setInfoWithId(\$id, \$data)*
  - `/social/save_info/id`: Updates existing information or adds new about the participant
- *function fetchTextsById(\$id)*
  - `/social/fetch_texts/id`: Returns all stored written text for this participant
- *function addTextToID(\$id, \$data, \$key = Null)*
  - `/social/add_text/id/timestamp`: Adds a new written text for a specific participant
- *function updateTextToID(\$id, \$data, \$key)*
  - `/social/update_text/id/timestamp`: Updates various data around a specific written text from a participant, for example extra features like frailty classification or events triggered

### 3.7 eCRF user authentication

The potential participant of the LingTester online tool is required to authenticate himself with the central FrailSafe database used by clinical partners known as Electronic Case Report Form (eCRF). This step is crucial as it ensures the profile consolidation of the participant throughout the different electronic platforms of the FrailSafe project and gives the ability to systematically gather and store the participants' data in the central FrailSafe database.

To support the authentication process (referred to as *Authentication sub-system* according to T6.2) a php library has been developed as an integrated part of the LingTester. The main methods and their corresponding eCRF API calls the library support are:

- *function getAuthToken(\$username,\$password)*
  - `/auth/v1/login`: Sends the user credentials to the eCRF server and if it succeeds to authenticate the user it returns an authentication token.

<sup>6</sup> <https://en.wikipedia.org/wiki/JSON>

- *function getEcrfid(\$username,\$authToken)*
  - */ecrfcore/v1/user/\$username/ecrfid: Returns the eCRF id of the participant for further use of his profile.*

After the successful authentication process of the participant, LingTester does not store any of the user credentials thus avoiding any security concerns.

### 3.8 Credentials Encryption

At the time being developed, the LingTester online tool as a part of the FrailSafe program has no intention to store any of the participants credentials relating to the available data services he authorizes us to use. However, some of the optional features the tool supports(as a final product), like participant's personal email account, created the need to implement a strong encryption layer in case the system has to store sensitive credential data.

In more detail, in order to avoid the current state of the art cryptanalysis techniques, sensitive LingTester data stored in the eCRF database are first encrypted with bcrypt. Bcrypt (Provos & Mazières, 1999) is an adaptive hash function based on the Blowfish symmetric block cipher cryptographic algorithm and introduces a work factor (also known as security factor), which allows you to determine how expensive the hash function will be. This work factor value determines how slow the hash function will be, means different work factor will generate different hash values in different time span, which makes it extremely resistant to brute force attacks. The php implementation of the library was selected as a part of our security layer as it was the most appropriate with the rest of the system.

### 3.9 User-flow

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.

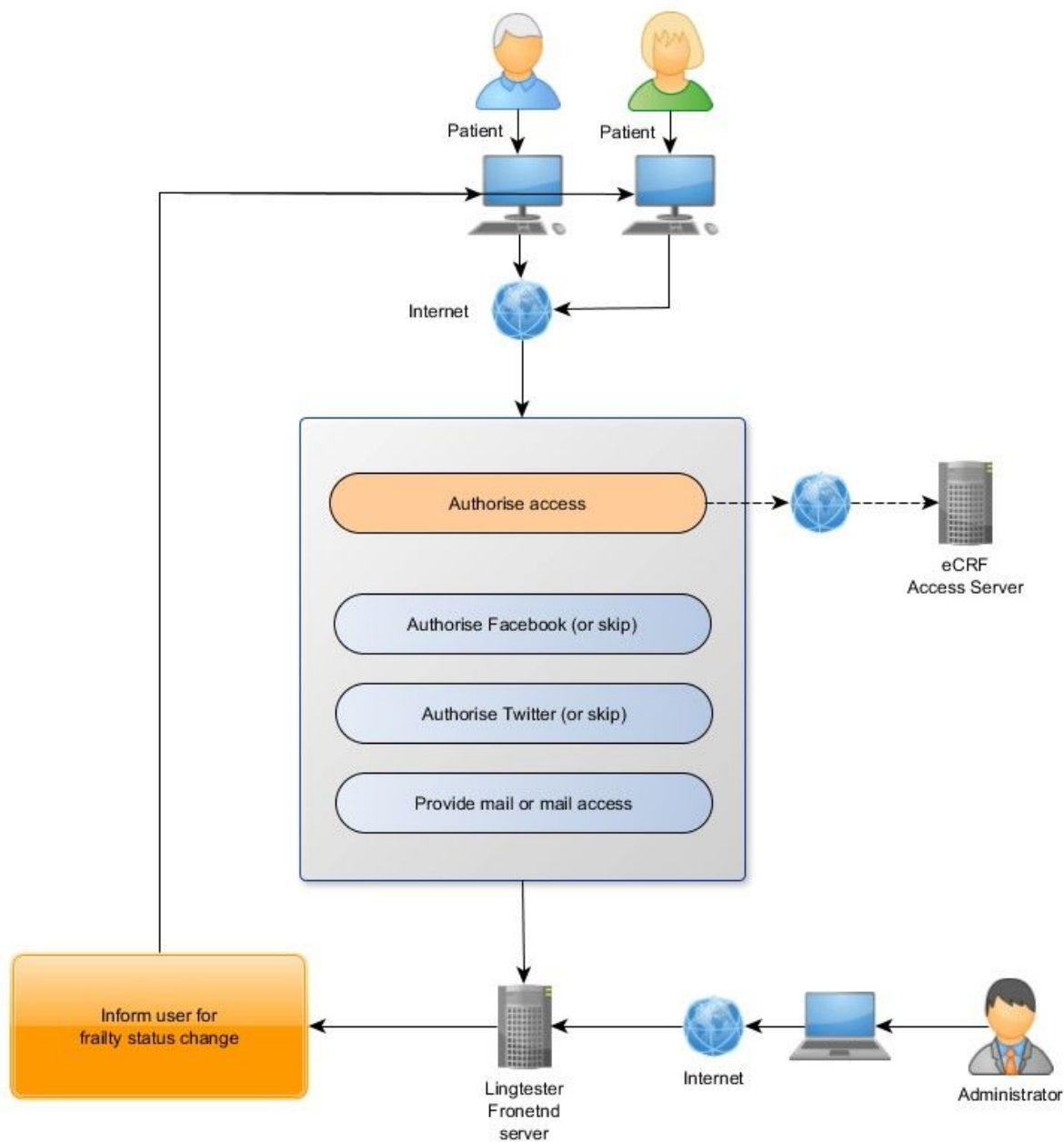


Figure 4: User flow

User (or participant 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”

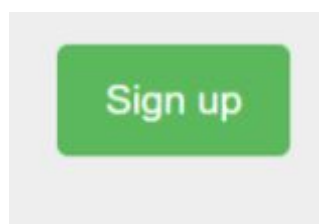




Figure 5: Step 1

- Step 2, eCRF authentication
  - User must put his eCRF username and password in order to proceed

### eCRF Participant Authentication

Your eCRF email:

Your password:

Submit

Figure 6: Step 2

- Step 3, 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.

Associate with facebook

Skip

Figure 7: Step 3

- Step 4, 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.

Associate with Twitter

Skip

Figure 8: Step 4

- Step 5, 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)

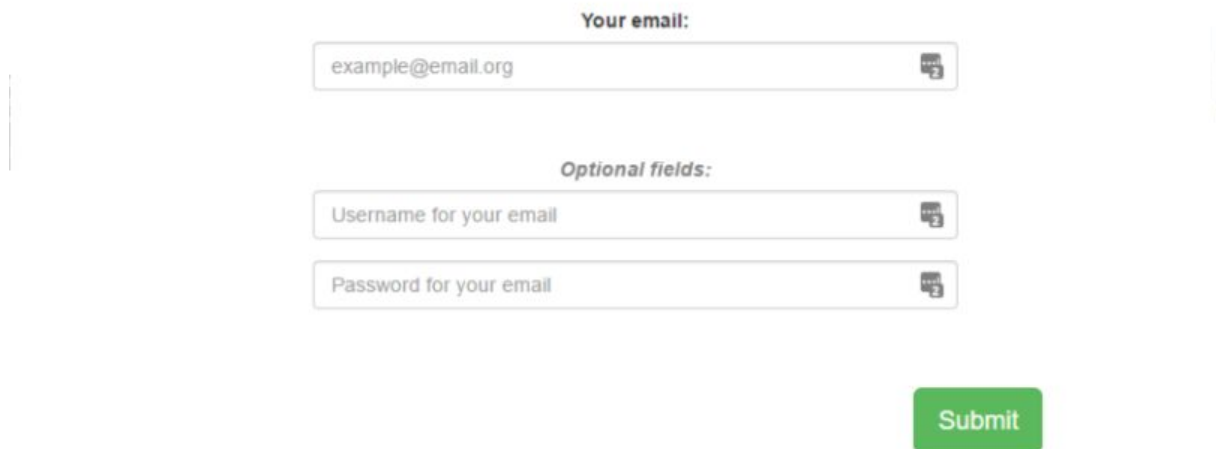


Figure 9: Step 5

- Step 5, final step
  - User concludes the process

### 3.10 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 overall behavior.

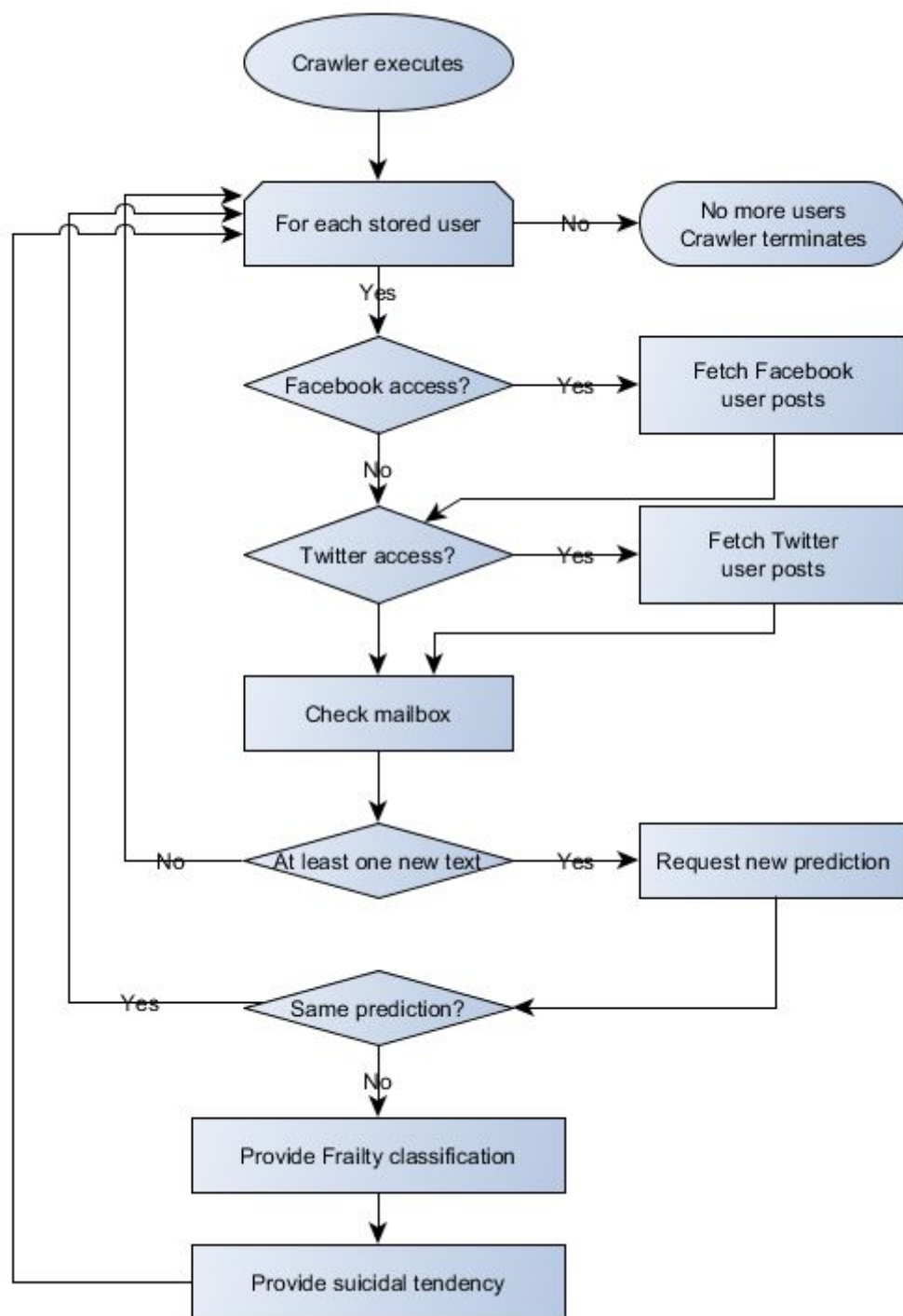


Figure 10: Crawler algorithmic procedure

## 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 11) 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 any 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.

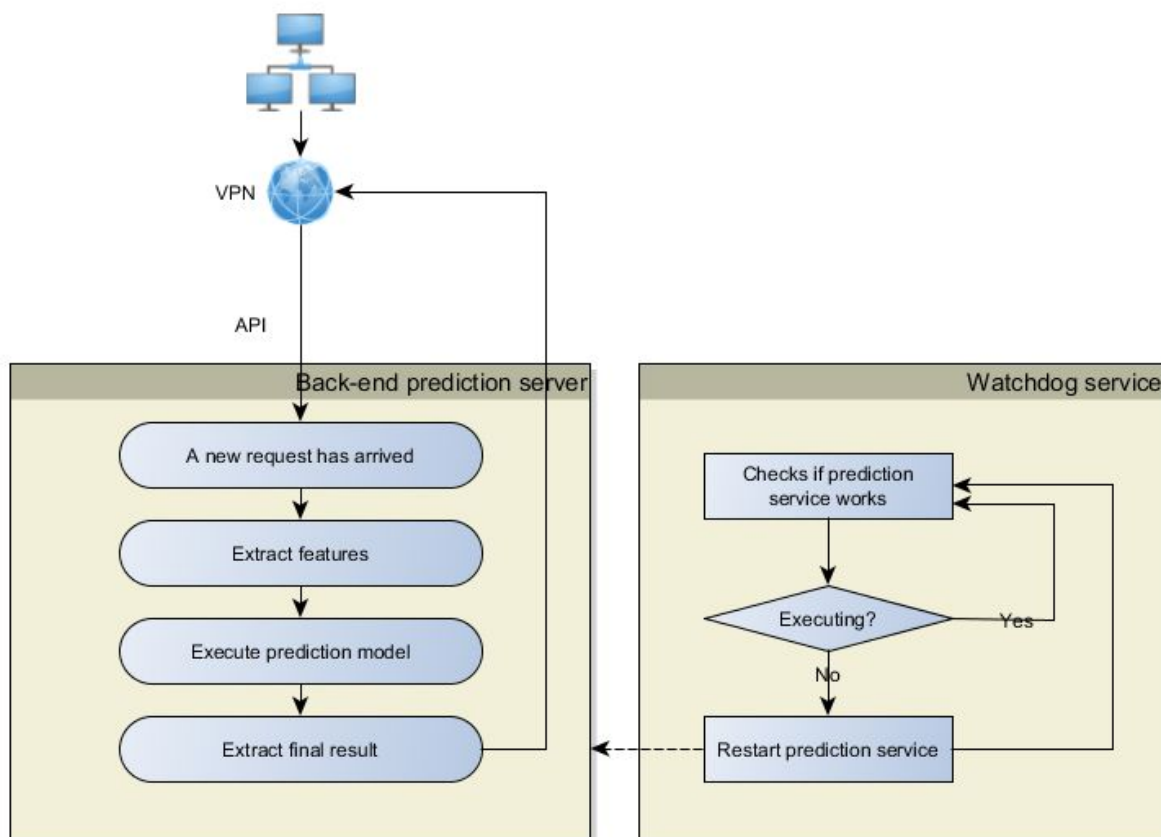


Figure 11: Backend architecture

## 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 initially 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 httpplib2`
  - Install httpplib2 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 Frailafe 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 the online LingTester tool is a large scale platform with a broad range of different technologies and applications cooperating to continuously monitor, record and analyse participants' data, it is considered necessary to deploy a series of tests and validate their corresponding results in relation with the offline results of the models obtained in **D4.13**. In this chapter, test results of different use case scenarios for the LingTester tool are presented. The exact process that was followed in order to obtain the test results is also explained.

The FrailSafe program partners with public health organizations, it was extremely difficult to test the tool with live participants mainly due to ethical reasons. In detail, as the doctors argued if the participants signed up with the online tool the software potentially could have cross-checked and reveal the identity of the participants, due to the use of social media accounts, although as it is explained the system filters and never stores any sensitive data from social media accounts. Nonetheless, this could raise serious ethical issues within the health organizations' competent committees and in the context of FrailSafe project this was not allowed.

To overcome this obstacle and safely run the platform testing, it was decided to use the already collected frailty real patient data that are already anonymized and can not potentially be cross-checked. In any way, these results are useful mainly to validate the correctness of the program flow, as the predictive model performance is already valuated in **D4.13**.

### 5.2 Participants

In order to run the test results for the online software tool, ten real FrailSafe participants were selected from the FrailSfe database. The participant selection was random without considering any of their mental, physical or language parameters, in order to conduct an almost real experiment with unbiased test assumptions. For each of the selected participants, one fake Facebook, Twitter or email account was created as a means to sign up with the LingTester online platform.

All user accounts are kept private with no social interaction. At the end of the software testing period, all the accounts including their data were removed. Each participant is referred by its user id.

The real participants' Frailty status along with their basic profile parameters like sex, year of birth, language are summarized in the next table.

eCRF ID	Service	Username	Year of birth	Sex	Language	Real Class
1045	facebook	user_1045@lingtester.fr ailsafe-project.cloud	1943	Male	Greek	Nonfrail
2049	facebook	user_2049@lingtester.fr ailsafe-project.cloud	1940	Female	Greek-Cypriot	Nonfrail
1082	facebook	user_1082@lingtester.fr ailsafe-project.cloud	1938	Male	Greek	Frail
3037	twitter	user_3037@lingtester.fr ailsafe-project.cloud	1934	Male	French	Prefrail
1060	email	user_1060@lingtester.fr ailsafe-project.cloud	1943	Male	Greek	Nonfrail
2006	twitter	user_2006@lingtester.fr ailsafe-project.cloud	1940	Male	Greek-Cypriot	Frail
2031	twitter	user_2031@lingtester.fr ailsafe-project.cloud	1936	Female	Greek-Cypriot	Prefrail
1018	facebook	user_1018@lingtester.fr ailsafe-project.cloud	1942	Female	Greek	Frail
3072	email	user_3072@lingtester.fr ailsafe-project.cloud	1933	Female	French	Nonfrail
1076	twitter	user_1076@lingtester.fr ailsafe-project.cloud	1931	Male	Greek	Prefrail

Table 1: List of participants

For each one of the above participants, text-posts were created and loaded to his account, on the service he was assigned to (Facebook, twitter or email).

In more detail, all text phrases found in the FrailSafe database corresponding to each participant's eCRF id were loaded as posts or emails. 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.



## 5.3 Classification model

The prediction model(Michalski, 2013) of the final online LingTester tool is based on the model that was presented in D4.13 with slight modifications to adjust to the online tool's restrictions and needs. In this subparagraph will be briefly presented for consistency. A table with the basic features it uses follows below.

Feature Names	Type - Extraction Method
<ul style="list-style-type: none"> <li>● transcript <ul style="list-style-type: none"> <li>○ yes</li> <li>○ no</li> </ul> </li> <li>● language <ul style="list-style-type: none"> <li>○ greek</li> <li>○ greek-cypriot</li> <li>○ french</li> </ul> </li> <li>● class <ul style="list-style-type: none"> <li>○ nonfrail</li> <li>○ prefrail</li> <li>○ frail</li> </ul> </li> <li>● data <ul style="list-style-type: none"> <li>○ Date of the submission for the transition study</li> </ul> </li> <li>● sex <ul style="list-style-type: none"> <li>○ male</li> <li>○ female</li> </ul> </li> <li>● do_you_consider_yourself_a_familiar_user_of_social_media <ul style="list-style-type: none"> <li>○ beginner</li> <li>○ less-familiar</li> <li>○ very-familiar</li> </ul> </li> <li>● family_status <ul style="list-style-type: none"> <li>○ married-or-in-a-relationship</li> <li>○ single</li> <li>○ divorced,</li> <li>○ widow</li> </ul> </li> <li>● habitation_zone <ul style="list-style-type: none"> <li>○ urban</li> <li>○ semi-urban</li> <li>○ rural</li> </ul> </li> <li>● have_you_changed_your_security_settings_in_social_media_in_order_to_protect_your_personal_data <ul style="list-style-type: none"> <li>○ yes</li> </ul> </li> </ul>	<b>Primitive</b> Rules & filters on eCRF API data

<ul style="list-style-type: none"> <li>○ no</li> <li>● year_of_birth</li> <li>● con_per_week <i>connections per week</i></li> <li>● twitter_follows <i>number if people user is following on Twitter</i></li> <li>● twitter_followers <i>number of followers on Twitter</i></li> <li>● fb_friends <i>number of friends on FB</i></li> </ul>	
<ul style="list-style-type: none"> <li>● text_length</li> <li>● number_of_sentences</li> <li>● number_of_words</li> <li>● number_of_words_per_sentence</li> <li>● text_entropy</li> </ul>	<b>Derived</b> Statistical Measures
<ul style="list-style-type: none"> <li>● desc_image_ENG_sentiment</li> <li>● desc_event_ENG_sentiment</li> <li>● prev_text_ENG_sentiment</li> </ul>	<b>Derived</b> Sentiment Analysis
<ul style="list-style-type: none"> <li>● desc_image_misspelled</li> <li>● desc_event_misspelled</li> <li>● prev_text_misspelled</li> </ul>	<b>Derived</b> Percent of misspelled words based on known vocabulary
<ul style="list-style-type: none"> <li>● tf-0</li> <li>● tf-1</li> <li>● ...</li> </ul>	<b>Derived</b> Term frequency – Inverse document frequency, after feature selection based on information gain
<ul style="list-style-type: none"> <li>● flesch_reading_ease</li> <li>● smog_index</li> <li>● flesch_kincaid_grade</li> <li>● coleman_liau_index</li> <li>● automated_readability_index</li> <li>● dale_chall_readability_score</li> <li>● difficult_words</li> <li>● linsear_write_formula</li> <li>● gunning_fog</li> </ul>	<b>Derived</b> Readability score
<b>Feature Names</b>	<b>Type - Extraction Method</b>

Table 2: List of features of the prediction model

As regards the deployed algorithm, a pre-trained model of the ensemble classifier VoteSRLR (Average probabilities voting between Simple Logistic, RotationForest, LMT, RandomForest) is used with its most important parameters optimized as follows.

Parameter Name	Description	Optimum Value
<b>Voting Algorithm</b>		
Combination Rule	The combination to rule used.	Average of Probabilities
<b>Simple Logistic</b>		
Error On Probabilities	Use error on the probabilities as error measure when determining the best number of LogitBoost iterations.	False
Weight Trim Beta	Set the beta value used for weight trimming in LogitBoost.	0.1
Heuristic Stop	LogitBoost is stopped if no new error minimum has been reached in the last Heuristic Stop iterations.	50
<b>RotationForest</b>		
Removed Percentage	The percentage of instances to be removed.	50
Projection Filter	The filter used to project the data.	Principal Components
Classifier	The base classifier to be used	J48
<b>LMT</b>		
Do Not Make Split Point Actual Value	If true, the split point is not relocated to an actual data value.	False
Weight Trim Beta	Set the beta value used for weight trimming in LogitBoost.	0.0
Fast Regression	Use heuristic that avoids cross-validating the number of Logit-Boost iterations at every node.	True
<b>RandomForest</b>		
Calc Out Of Bag	Whether the out-of-bag error is calculated.	False
Max Depth	The maximum depth of the tree, 0 for unlimited.	0

Table 3: Algorithmic parameters of the prediction model

## 5.3 Suicidal model tests & results

One of the main purposes of the online tool is the constant monitoring of the participants' written text, through all the available data sources that he has authorized as to monitor, for early signs of suicidal dispositions.

The detailed study of the machine learning model is subject of **D4.13**. In brief, the constructed model takes in account a number of general sentence statistical features like text entropy, readability scores, misspellings as well as more specific to the problem features like sentiment analysis scores and term frequencies for words like: cry, death, depression, kill, sorrow e.t.c.

The final tuned machine learning model is a Sequential Minimal Optimization (SMO) (Platt, 1998) model which is based on the very well known support vector machine model. Among the other compared models, it managed to outperform them scoring an overall accuracy of 71.15% correctly classified cases.

In the next subparagraph an algorithm explaining the Testing process will be presented. The online LingTester tool is capable of analyzing a user's texts and produce a probability is analogous to the suicidal tendency the user's texts reveal. The next image, is a direct output from the console log of the online tool and presents us in brief those probabilities for the test users.

```
Prediction for instance: 0 is: non-suicide suicidal-percentage: 0.2078
Prediction for instance: 1 is: non-suicide suicidal-percentage: 0.3524
Prediction for instance: 2 is: non-suicide suicidal-percentage: 0.3538
Prediction for instance: 3 is: non-suicide suicidal-percentage: 0.4802
Prediction for instance: 4 is: non-suicide suicidal-percentage: 0.3102
Prediction for instance: 5 is: non-suicide suicidal-percentage: 0.3791
Prediction for instance: 6 is: non-suicide suicidal-percentage: 0.1976
Prediction for instance: 7 is: non-suicide suicidal-percentage: 0.1363
Prediction for instance: 8 is: non-suicide suicidal-percentage: 0.2186
Prediction for instance: 9 is: non-suicide suicidal-percentage: 0.2800
Prediction for instance: 10 is: non-suicide suicidal-percentage: 0.2306
```

Image 1. Command line results

The fields 'suicidal-percentage' seen in the above image correspond to the probabilities mentioned before. It very important to be understood that these probabilities are only a measurement of the current text and can only be used as an alarm for the system operator. The final decision should clearly be made by an expert, usually doctor, on the subject. Probabilities less than 0.70 (or 70%) can be safely ignored as the text is judged as not clearly suicidal.

## 5.5 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 constructed user profiles was utilized in the following process.

### LingTester online - Testing process

---

#### 1. For each user profile:

- a. Create a private account according to the Service assigned to him from table 1 (Facebook or Twitter or email).
- b. Load all his corresponding phrases from the pool of the participants' phrases according to his assigned eCRF id, based on the dataset obtained from **D4.13**, as provided by the clinical groups.
- c. Sign up the user to the LingTester online platform

#### 2. Expect the platform to analyze the user profile asynchronously and collect the frailty report from the LingTester tool.

#### 3. Repeat steps (1) and (2) for 10 times

#### 4. Evaluate the assigned users' class with the reported users' class.

---

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

eCRF ID	Collected Phrases	Real Class
1045	<ol style="list-style-type: none"> <li>1. Η αστάθεια του σκαμνι πάνω στο οποίο στηρίζεται το παιδί Το ξεχείλισμα του νεροχύτη</li> <li>2. Το ότι πέτυχα 3ος στο Πανεπιστήμιο.</li> </ol>	Nonfrail
2049	<ol style="list-style-type: none"> <li>1. Ένα αγόρι να στέκεται σε καρεκλάκι που δεν φαίνεται να στηρίζεται καλά στο πάτωμα κ' να προσπαθεί να πάρει μπισκότο. Το κορίτσι να ζητά μπισκότο, από το αγόρι. Η γυναίκα, που φαίνεται να είναι η μητέρα τους είναι «στό κόσμο της» τρέχουν τά νερά από τή βούρνα, σκουπίζει τα πιάτα, αγνοεί τα παιδιά, για να τά βοηθήσει. Γενικά, μου δημιουργείται η αίσθηση τού ότι δεν υπάρχει επικοινωνία</li> </ol>	Nonfrail

	<p>μεταξύ τους. Λείπει τό αίσθημα τής έγνοιας κ' τού ενδιαφέροντος στίς ανάγκες τών άλλων.</p> <p>2. Μια ξαφνική εγχείρηση καρδιάς, ατόμου άμεσα συνδεδεμένου μαζί μου με «ταρακούνησε».</p> <p>Σοβαρή εγχείρηση, με μικρές πιθανότητες επιτυχίας. Στή διάρκεια της εγχείρησης και στις δυό μέρες που ακολούθησαν καθοριστικές στη γνώση γιά τό αποτέλεσμα τής επέμβασης έκαμα αναδρομή στα τής «σχέσης μου», μέ τό άτομο αυτό. Έκαμα «τίμια» αυτοκριτική, ωσάν να ζητούσα συγνώμη από το άτομο αυτό, γιά όταν κ' άν έφταιξα. Όταν ξύπνησε κ' ήταν καλά ένοιωσα ανακούφιση, παράλληλα δέ κ' μιά απονεχοποίηση, έστω κ' άν ήμουν αυστηρή με τον εαυτό μου.</p>	
1082	<p>1. στενοχωριέμαι λόγω του ότι δεν είμαι στη δουλειά μου. Την αγαπώ και την έκανα από 10-11 ετών. Μανάβης ήμουν. Εκεί θέλω να γυρίσω!</p> <p>2. Μια οικογένεια είναι αυτή. Η μαμά και τα παιδιά.</p>	Frail
3037	<p>1. Souvenir de voyage La marche sur le Chemin de Compostelle est le voyage qui m'a le plus profondément marqué au cours de ces dernières années. Entrepris en 2014, à l'occasion de mes 80 ans, il s'est déroulé en plusieurs étapes</p> <p>2. Une cuisine bien tranquille La cuisine de Sylvia est agréable et bien aménagée. Une large fenêtre donnant sur la cité permet de se distraire tout en vaquant aux travaux domestiques. De vastes placards encastrés, un large plan de travail, le grand espace disponible permettent à la jeune maîtresse de maison de se sentir à l'aise. Mais aujourd'hui, c'est la catastrophe. Sylvia vient de terminer sa vaisselle et commence à l'essuyer sans se rendre compte qu'elle a laissé couler le robinet. L'évier est en train de déborder largement sur le sol et Sylvia, sans doute perdue dans ses rêves, n'y a même pas prêté attention. Tournant le dos à ses enfants, elle ne s'est pas aperçue non plus du danger qui les guette. Voulant gentiment l'aider ils se sont organisés, Christine passe les assiettes à Vincent et celui-ci, juché sur un haut tabouret, les range dans le placard du haut. Mais il vient de perdre l'équilibre et va faire une chute dangereuse en risquant de s'assommer contre le bord du plan de travail...</p>	Prefrail
1060	<p>1. Χαίρωμαι με πολλά που θυμαμαι</p> <p>2. Μια κυρία στην κουζίνα της έχει πλυμηρίσει ο νεροχύτης, η κουζίνα έχει ένα παράθυρο με κουρτίνες</p> <p>Ένα κοριτσάκι παρακολουθεί ένα αγόρι που είναι πάνω σε ένα σκαμπό και πάει να πέσει από αυτό, πώς προσπαθεί να πάρει κάτι από το ριφι</p>	Nonfrail
2006	<p>1. Ήταν Πάσχα 18 Απριλίου 1971. Την γυναίκα μου την έπιασαν οι πόνοι του τοκετού. Την μετέφερα στην κλινική για την γέννα. Αφού πέρασε περίπου μισή ώρα βγαίνει ο γιατρός από το μαιευτήριο και μου λέει. « έχουμε δύσκολο τοκετό και</p>	Frail

	<p>πρέπει να μου πεις ποιος θέλει να ζήσει ή μάνα ή το παιδί;» Σοκαρίστηκα, έπρεπε να πάρω απόφαση ποιός θα πεθάνει και ποιός θα ζήσει. Σίγουρα γιατρέ τους θέλω και τους δύο. Ο Θεός βοήθησε και έζησαν και οι δύο. Το παιδί ήταν αγοράκι. Το παιδί το χάσαμε όμως αργότερα σε αυτοκινητικό δυστύχημα στα 21 του χρόνια στις 28/11/1992.</p> <p>2. Εισ την πιο πάνω εικόνα βλέπω τρία άτομα μέσα σε μια κουζίνα. Προφανώς πρόκειται για μητέρα και τα δύο παιδιά της. Ένα αγόρι και ένα κορίτσι. Η μητέρα στέκεται μπροστά στο νεροχύτη φορώντας την ποδιά της νοικοκυράς και σκουπίζει τα πιάτα. Δεν πρόσεξε όμως ότι ο νεροχύτης ξεχείλισε και ότι τα νερά έτρεχαν στο πάτωμα, βρέχοντάς της τὰ παπούτσια. Πίσω από τον νεροχύτη υπάρχει παράθυρο με κουρτίνες. Η μικρή περιμένει, ενώ καπνίζει τσιγάρο, από τον αδελφό της να της δώσει, πιθανώς μπισκότα, από το πάνω ντουλάπι της κουζίνας. Όμως το σκαμνάκι που χρησιμοποιούσε έγειρα με κίνδυνο να πέσει κάτω.</p>	
2031	<p>1. No answer</p> <p>2. Εδώ βλέπω μια οικόγένεια που θέλοντας νά κάνει δουλιές κάνει ζημιές. Η μητέρα σκουπίζει τα πιάτα. Ο γιός προσπαθή νά πιάση κάτι από το ραφι καί θα πέση κατω, τό κορίτσι είναι τό πιο φρονιμο.</p>	Prefrail
1018	<p>1. Βλέπω το αγόρι και παίρνει cookies από της μαμάς, δίνει και στην αδερφή του. αλλά θα πέσει στο πάτωμα. Η μητέρα πλένει τα πιάτα και τα σκουπίζει.</p> <p>2. Τα παιδιά μου, τα εγγόνια μου</p>	Frail
3072	<p>1. No answer</p> <p>2. je vois une femme faire la vaisselle, l'évier déborde d'eau. un garçon monté sur un tabouret, celui-ci bascule. Il prend un gâteau dans le buffet et le tend à une fillette</p>	Nonfrail
1076	<p>1. Τώρα στο τέλος μήνα που αλλάζω χρόνο στη ζωή μου. Τα γενέθλιά μου!</p> <p>2. Εδώ βλέπουμε τούτο δω και τούτο, δεν βλέπω και καλά. Ζώα δεν βλέπω, δεν έχω και γυαλιά γιατί έχω καταρράχτη και περιμένω να κάνω χειρουργείο.</p>	Prefrail

Table 4: Test input

After the algorithm execution, ten frailty status predictions were obtained one for each of the user profiles. At this point we recall from **D4.13** that the prediction accuracy of VoteSRLR was calculated at 63.64% correctly classified instances on the three class frailty classification problem (nonfrail, prefrail, frail). The following table presents the frailty statuses the online LingTester tool recorded in the eCRF database after its asynchronous analysis of the ten signed up profiles.

eCRF ID	Predicted Class	Model Prediction Probability%	Real Class
1045	Nonfrail	89.21	Nonfrail
2049	Nonfrail	78.47	Nonfrail
1082	Frail	99.20	Frail
3037	Prefrail	61.24	Prefrail
1060	Nonfrail	92.32	Nonfrail
2006	Frail	99.8	Frail
2031	Frail	57.34	Prefrail
1018	Frail	69.23	Frail
3072	Nonfrail	84.32	Nonfrail
1076	Frail	50.42	Prefrail

Table 5: Prediction test results

By observing the above table, we extract the number of Correctly classified instances as 8 and the number of Incorrectly classified instances as 2 thus calculating a classification Accuracy of 80%. By comparing the accuracy of 80% with the cross-validated accuracy 63.64% of the model obtained in **D4.13** we measure an approximate difference of 16%. This is not without reason, we have to consider that the test set was relatively small compared to the one used in **D4.13**. Any subset of the full test set could have given a little better or worse accuracy compared to the full test set and this was something that it was expected. We have to make clear that this experiment was ran solely to validate the correctness of the online LingTester program flow and not to test the accuracy of the prediction model, which is a goal of **D4.13**.

## 5.6 Discussion of the results

The results we obtained were generally in line with the FrailSafe dataset. The overall accuracy was 80% 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 **D4.13** 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 obtain as good results as good as the integrated model can produce. The LingTester online platform is a useful tool that can be used in cooperation with clinical experts. It can provide a degree of mental status monitoring for the participants and produce a number of



alarm indicators. As every medical technology, it should be used with seriousness and caution.

## 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.
- Provos, Niels; Mazières, David; Talan Jason Sutton 2012 (1999). "A Future-Adaptable Password Scheme". Proceedings of 1999 USENIX Annual Technical Conference: 81–92.

## 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: 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

### 9.2 Frontend

#### 9.2.a Requests Router

File: catalog/view/main.php

```

1.  <!DOCTYPE html>
2.  <html>
3.      <head>
4.          <meta charset="utf-8">
5.          <meta http-equiv="X-UA-Compatible" content="IE=edge">
6.          <meta name="viewport" content="width=device-width, initial-scale=1">
7.
8.          <link href="image/favicon.ico" rel="shortcut icon" />
9.          <title>FrailSafe Online System</title>
10.
11.         <!--css-->
12.         <link href="catalog/lib/bootstrap/css/bootstrap.min.css" rel="stylesheet" />
13.         <link href="catalog/view/stylesheet/jumbotron-narrow.css" rel="stylesheet" />
14.         <link href="catalog/view/stylesheet/index.css" rel="stylesheet" />
15.
16.         <!--js-->
17.         <script type="text/javascript" src="catalog/lib/jquery/js/jquery-3.2.0.min.js"></script>
18.         <script type="text/javascript" src="catalog/lib/bootstrap/js/bootstrap.min.js"></script>
19.         <script type="text/javascript" src="catalog/view/js/index.js"></script>
20.
21.     </head>
22.     <body>
23.
24.         <div class="container">
25.             <?php
26.                 include_once 'catalog/view/code/header.php';
27.             ?>
28.
29.             <?php
30.                 if(isset($_REQUEST['home']))
31.                     include_once 'catalog/view/code/home.php';
32.                 elseif(isset($_REQUEST['step1']))
33.                     include_once 'catalog/view/code/step1.php';
34.                 elseif(isset($_REQUEST['step2']))
35.                     include_once 'catalog/view/code/step2.php';
36.                 elseif(isset($_REQUEST['step3']))
37.                     include_once 'catalog/view/code/step3.php';

```

```

38.         elseif(isset($_REQUEST['success']))
39.             include_once 'catalog/view/code/success.php';
40.         elseif(isset($_REQUEST['about']))
41.             include_once 'catalog/view/code/about.php';
42.         elseif(isset($_REQUEST['contact']))
43.             include_once 'catalog/view/code/contact.php';
44.         else
45.             include_once 'catalog/view/code/home.php';
46.     ?>
47.
48.
49.     <?php
50.     include_once 'catalog/view/code/footer.php';
51.     ?>
52.
53. </div>
54. <!-- /container -->
55.
56. </body>
57. </html>

```

### 9.2.b Registration

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

```

1. <?php
2.     session_start();
3.     // config and whatnot
4.     $config = dirname(__FILE__) . '/../hybridauth/config.php';
5.     require_once( "../hybridauth/Hybrid/Auth.php" );
6.
7.     $user_data = NULL;
8.     $returnUrl = urldecode($_GET['returnurl']);
9.     //echo $_GET['returnurl'];
10.    //return;
11.
12.    // try to get the user profile from an authenticated provider
13.    try{
14.        $hybridauth = new Hybrid_Auth( $config );
15.
16.        // selected provider name
17.        $provider = @ trim( strip_tags( $_GET["provider"] ) );
18.
19.        // check if the user is currently connected to the selected provider
20.        if( ! $hybridauth->isConnectedWith( $provider ) ){
21.            // redirect him back to login page
22.            header( "Location: login.php?error=Your are not connected to $provider or your
                session has expired" );

```

```

23.     }
24.
25.     // call back the requested provider adapter instance (no need to use authenticate() as
    we already did on login page)
26.     $adapter = $hybridauth->getAdapter( $provider );
27.
28.     // grab the user profile
29.     $user_data = $adapter->getUserProfile();
30.     //session_destroy();
31.     //session_start();
32.     include_once(' ../../../../config.php');
33.     include_once(' ../../../../catalog/controller/dbOperator.php');
34.     $dbOp = new dbOperator($dbHost,$dbUsername,$dbPassword,$dbName);
35.
36.     $user = $dbOp->query("Select * from user where
    user".$provider."Email='".$user_data->email.'" LIMIT 1");
37.     //print_r($user);
38.     if(!$user->num_rows){
39.         echo 'not registered';
40.         $id = register($dbOp,$user_data,$provider);
41.     }else
42.         //$id = $user[0]['idUser'];
43.         $id = $user->row['idUser'];
44.
45.     if(strpos($returnUrl, "step1"))
46.         $returnUrl = str_replace("step1", "step2", $returnUrl);
47.     elseif(strpos($returnUrl, "step2"))
48.         $returnUrl = str_replace("step2", "step3", $returnUrl);
49.
50.     if(!isset($_SESSION['idUser'])){
51.         $_SESSION['idUser'] = $id;
52.
53.         //$urlParameter = '';
54.         //header("Location:
    ".((strpos($returnUrl, '?'))?urlencode($returnUrl).'&'.$urlParameter:urldecode($returnUrl).'?'.$u
    rlParameter) );
55.         header("Location: ".urlencode($returnUrl));
56.     }else
57.         header("Location: ".urlencode($returnUrl));
58.
59. }
60. catch( Exception $e ){
61.     // In case we have errors 6 or 7, then we have to use Hybrid_Provider_Adapter::Logout()
    to
62.     // Let hybridauth forget all about the user so we can try to authenticate again.
63.
64.     // Display the received error,
65.     // to know more please refer to Exceptions handling section on the userguide
66.     switch( $e->getCode() ){

```

```

67.         case 0 : echo "Unspecified error."; break;
68.         case 1 : echo "Hybriauth configuration error."; break;
69.         case 2 : echo "Provider not properly configured."; break;
70.         case 3 : echo "Unknown or disabled provider."; break;
71.         case 4 : echo "Missing provider application credentials."; break;
72.         case 5 : echo "Authentication failed. "
73.             . "The user has canceled the authentication or the provider refused the
connection.";
74.         case 6 : echo "User profile request failed. Most likely the user is not connected "
75.             . "to the provider and he should to authenticate again.";
76.             $adapter->logout();
77.             break;
78.         case 7 : echo "User not connected to the provider.";
79.             $adapter->logout();
80.             break;
81.     }
82.
83.     echo "<br /><br /><b>Original error message:</b> " . $e->getMessage();
84.
85.     echo "<hr /><h3>Trace</h3> <pre>" . $e->getTraceAsString() . "</pre>";
86. }
87.
88.
89. function register($dbOperator,$userData,$provider=''){
90.     $dbOp = $dbOperator;
91.     $user_data = $userData;
92.
93.     if(!isset($_SESSION['idUser'])){
94.         if($dbOp->query("INSERT INTO `user`(`idUser`, `user".$provider."Id`,
`user".$provider."Email`) VALUES ('','$user_data->identifier','$user_data->email.'"))
95.             $id = $dbOp->getLastId();
96.     }elseif(!empty($_SESSION['idUser'])){
97.         $dbOp->query("UPDATE user SET `user".$provider."Id`='$user_data->identifier',
`user".$provider."Email`='$user_data->email.'" WHERE idUser=".$_SESSION['idUser']);
98.         return $_SESSION['idUser'];
99.     }
100.     return $id;
101. }
102. ?>

```

## 9.3 Backend

### 9.3.a Crawler main loop

File: crawler/index.php

```
<?php

include_once '../config.php';
$auth_config = include_once
'../catalog/lib/hybridauth/hybridauth/config.ph
p';
include_once
'../catalog/controller/dbOperator.php';
include_once '../catalog/controller/api.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();

print str_repeat("\n", 10);
print '<pre>';

print date('m/d/Y H:i:s') . "\n";

$all_ids = getALLIDs();
$all_user_objs = array();
foreach ($all_ids as $user_id) {
    $ret = getInfoById($user_id);
    if (empty($ret))
        continue;

    $all_user_objs[$user_id] = $ret;
}

//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']['us
ername'];

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

//Set who the message is to be sent from
$mail->setFrom($auth_config['providers']['Googl
e']['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_time
```



```

Line.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);

// // $all_users = $dbOp->query('Select * from
user;');
// // To move all users from MySQL to API
// foreach ($all_users->rows as $user) {
//     // print_r(removeID(8000 +
$user['idUser']));
// }
// continue;

// $one_user = (array)$user;
// unset($one_user['idUser']);
// setInfoWithId(8000 + $user['idUser'],
$user);
// }

// Check all Facebook texts
$start = microtime(True);
foreach ($all_user_objs as $user_id => $user) {
    // print $user_id . ' ' .
print_r(fetchTextsByID($user_id));
    // $users_with_new_texts[ $user['idUser'] ] =
1;
    // $user_id = $user['idUser']

    // Facebook
    if (!array_key_exists('userFacebookId',
$user) || empty($user['userFacebookId']))
        continue;

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

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

    if (empty($result))
        continue;

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

    if (empty($decoded))
        continue;

    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;');
        $existing_post = fetchTextsByID($user_id,
$utime);

        if (!empty($existing_post))
            continue;

        $text = anonymizeText($value['message']);
        addTextToID($user_id, array('text' =>
$text, 'source' => 'facebook'), $utime);

        // Keep it handy
        $users_with_new_texts[ $user_id ][$utime] =
$text;
    } // foreach ($decoded['data'] as $value)
} // foreach ($all_ids as $user_id => $user)
$end = microtime(True);
print sprintf("Checked facebook after
%.2fsec\n", $end - $start);

// Check all Twitter users
$start = microtime(True);
foreach ($all_user_objs as $user_id => $user) {
    if (empty($user['userTwitterId']))
        continue;

    $ret = $twitter->setGetfield('?user_id=' .
$user['userTwitterId'])
->buildOAuth($twitterURL, 'GET')
->performRequest();

    if (!$ret || !is_array($ret))
        continue;

    foreach ($ret as $tweet) {
        $utime = strtotime($tweet->created_at);
        $lang = $tweet->lang;

        $existing_post = fetchTextsByID($user_id,
$utime);
        if (!empty($existing_post))
            continue;

        $text = $tweet->text;
        $text = anonymizeText($text);
        addTextToID($user_id, array('text' =>
$text, 'source' => 'twitter'), $utime);

        // Keep it handy
        $users_with_new_texts[ $user_id ][$utime] =
$text;
    } // foreach ($ret as $tweet)
} // foreach ($all_users as $user)
$end = microtime(True);
print sprintf("Checked Twitter after
%.2fsec\n", $end - $start);

$new_emails = getEmails(
    $auth_config['providers']['Google']['connection
'],

```

```

$auth_config['providers']['Google']['keys']['userna
ername'],

$auth_config['providers']['Google']['keys']['pa
ssword']
);

$start = microtime(True);
if (!empty($new_emails)) {
    // Topikos pinakas gia na kserw pou paei pou
    efkola
    $email_per_user = array();
    foreach ($all_user_objs as $user_id => $user)
    {
        foreach (array('userEmail',
            'userTwitterEmail', 'userFacebookEmail') as
            $key) {
            if (!array_key_exists($key, $user) ||
                empty($user[$key]))
                continue;

            $email_per_user[ strtolower($user[$key])
            ] = $user_id;
        } // foreach (array('userEmail') as $key)
    } // foreach ($all_user_objs as $user_id =>
        $user)

    foreach ($new_emails as $email_number =>
        $email_data) {
        $from = strtolower($email_data['from']);
        if (!array_key_exists($from,
            $email_per_user)) {
            // Den dexomaste email apo asxetous
            // .. 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 our system.
            Please visit
            https://lingtester.frailsafe-project.cloud/ and
            follow the steps.';
            $mail->send();
            continue;
        } // if (!array_key_exists($from,
            $email_per_user))

        $utime = $email_data['utime'];

        $existing_post = fetchTextsByID($user_id,
            $utime);
        if (!empty($existing_post))
            continue;

        $text = $email_data['message'];
        $text = anonymizeText($text);
        addTextToID($user_id, array('text' =>
            $text, 'source' => 'email'), $utime);

        // Keep it handy
        $users_with_new_texts[ $user_id ][ $utime ] =
            $text;
    } // foreach ($new_emails as $email_number =>

```

```

$email_data)

    print_r($email_per_user);
} // if (!empty($new_emails))
$end = microtime(True);
print sprintf("Checked inbox after %.2fsec\n",
    $end - $start);

if (count($users_with_new_texts) <= 0) {
    print 'No users with new text<br>';
    return;
} // if (count($users_with_new_texts) <= 0)

print "Users with new text\n";
print_r($users_with_new_texts);

foreach ($users_with_new_texts as $user_id =>
    $utimes) {
    $max_suicidal = 0;
    foreach ($utimes as $utime => $text) {
        // Elegxos suicidal
        $text_obj = fetchTextsByID($user_id,
            $utime);
        if (empty($text_obj) ||
            empty($text_obj['text']))
            continue;

        // $text = $text_obj['text'];
        $suicidal_obj =
            getPredictionSuicidal($text, $backend);

        print "Suicidal object\n";
        print_r($suicidal_obj);

        if (empty($suicidal_obj) ||
            !array_key_exists('suicidal_percentage',
                $suicidal_obj)) {
            print "Suicidal object INVALID
            retrieved\n";
            print_r($suicidal_obj);
            continue;
        }

        $suicidal_percent =
            floatval($suicidal_obj['suicidal_percentage']);
        $max_suicidal = max($suicidal_percent,
            $max_suicidal);

        // Update everything
        $must_save = False;
        foreach (array('various_info', 'features')
            as $key) {
            if (!array_key_exists($key,
                $suicidal_obj))
                continue;

            foreach ($suicidal_obj[$key] as $tag_key
                => $tag_value) {
                $save_key = sprintf('suicidal_%s_%s',
                    $key, str_replace('-', '_', $tag_key));
                if (array_key_exists($save_key,
                    $text_obj) && $text_obj[$save_key] ==
                    $tag_value)
                    continue;

                $text_obj[$save_key] = $tag_value;
                $must_save = True;
            }
        }
    }
}

```

```

        } // foreach ($suicidal_obj[$key] as
$tag_key => $tag_value)
        } // foreach (array('various_info',
'features') as $key)

        if ($must_save) {
            print "Text object retrieved\n";
            print_r($text_obj);
            updateTextToID($user_id, $text_obj,
$utime);
        } // if ($must_save)

        $msg = sendSocialAlert($user_id, $utime,
array('suicidal_percent' =>
$suicidal_percent));
        var_dump($msg);
    } // foreach ($utimes as $utime)

    if ($max_suicidal > $threshold_to_send_email
&& !empty($inform_for_suicidal)) {
        // Must inform somebody
        $mail->ClearAddresses();
        $mail->addAddress($inform_for_suicidal);
        $mail->Subject = 'FrailSafe: Lintester,
suicidal prediction';
        $mail->Body = sprintf('User %d has showed
%.1f%% suicidal behavior.', $user_id,
$max_suicidal * 100);
        $mail->send();
        continue;
    } // if ($max_suicidal >
$threshold_to_send_email)
} // foreach ($users_with_new_texts as $user_id
=> $utimes)

foreach ($users_with_new_texts as $user_id =>
$utimes) {
    // foreach ($utimes as $utime) {
    // }
    // This user has some new text
    // .. take all available 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
    $ret = getPredictionFrailty($old_texts,
$new_texts, $backend);
    // Backend issue?
    if (!$ret) {
        print 'Unable to get prediction<br>';

```

```

        continue;
    }

    $data = json_decode($ret);
    // Backend issue?
    if (!$data || !isset($data->prediction)) {
        print 'Unable to decode prediction<br>';
        print 'Raw data: ' . $ret . '<br>';
        continue;
    }

    // Model issue?
    if (!in_array($data->prediction,
array('frail', 'prefrail', 'nonfrail'))) {
        print 'Unknown prediction: ' .
$data->prediction . '<br>';
        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 = '';
    $show_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 $value)
    {
        if ($show_many_times == 0 || $new_prediction
== $value['status']) {
            $show_many_times += 1;
            $new_prediction = $value['status'];
        } // if ($show_many_times == 0 ||
$new_prediction == $value['status'])
        } // foreach ($prediction_history->rows as
$value)

    // This doesn't qualify for notification yet
    if ($show_many_times <
SAME_PREDICTIONS_IN_A_RAW || $new_prediction ==
'') {
        print 'Same prediction in a raw<br>';
        continue;
    }

    // No need to notify
    if (!in_array($new_prediction,
NOTIFY_WITH_TAGS)) {
        print 'Prediction not in the ones I care ('
. implode(', ', NOTIFY_WITH_TAGS) . ')<br>';
        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) {
    print 'Too soon to send again email<br>';
    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';
$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);
}
else {
    print 'Unable to send email to ' .
    $user_row->row['userEmail'] . '<br>';
}

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

### 9.3.b Text Parser

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

from flask import Flask
from flask import request

import os
import offline_parser
import json
import time
import langdetect
import pickle
import re

app = Flask(__name__)

def identifyLang(Text):
    ret = langdetect.detect(Text)
    prob = langdetect.detect_langs(Text)

    if ret == 'el':
        return ['greek', prob]

    if ret == 'fr':
        return ['french', prob]

    return ['english', prob]

def create_frailsafe_arff(oldText, newText,
relation = 'frailsafe_111'):
    """Create arff for WEKA with all features
    available"""
    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 = ('nonfrail', '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_entrop
y('',
'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, perc] = 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))
            row.append(str(0))

        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_misspelling_score(tex
t, lang = clang)
            row.append(str(ret))

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

    f.close()

def create_suicidal_arff(checkText, relation =
'frailsafe_222'):
    """Create arff for WEKA with all features
available"""
    out = []
    out.append('@RELATION %s' % relation)
    out.append('')

        out.append('@ATTRIBUTE %s %s' %
(offline_parser.get_feature_length('',
'title'),
offline_parser.get_feature_length('', 'type')))
        out.append('@ATTRIBUTE %s %s' %
(offline_parser.get_feature_number_of_sentences
('',
'title'),
offline_parser.get_feature_number_of_sentences(
'', 'type')))
        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_words_per_sentence(
'',
'title'),
offline_parser.get_feature_words_per_sentence(
'', 'type')))
        out.append('@ATTRIBUTE %s %s' %
(offline_parser.get_feature_text_shannon_entrop
y('',
'title'),
offline_parser.get_feature_text_shannon_entropy
('', 'type')))

```

```

read_scores = []
read_scores.append('flesch_reading_ease')
read_scores.append('smog_index')
read_scores.append('flesch_kincaid_grade')
read_scores.append('coleman_liau_index')

read_scores.append('automated_readability_index')

read_scores.append('dale_chall_readability_score')
read_scores.append('difficult_words')
read_scores.append('linsear_write_formula')
read_scores.append('gunning_fog')
for attr in read_scores:
    out.append('@ATTRIBUTE %s %s' % (attr,
'real'))

    out.append('@ATTRIBUTE %s %s' %
('sentiment', 'real'))
    out.append('@ATTRIBUTE %s %s' %
('misspelled', 'real'))

# TFIDF
words_to_check = [
    'commit',
    'cry',
    'death',
    'depression',
    'die',
    'feel',
    'feeling',
    'felt',
    'grief',
    'happiness',
    'heart',
    'it',
    'kill',
    'me',
    'money',
    'myself',
    'pain',
    'sad',
    'sadness',
    'sex',
    'sorrow',
    'suicide',
    'suicides',
    'tears',
    'was',
    'work',
    'writing',
    'your',
]

for i in range(len(words_to_check)):
    out.append('@ATTRIBUTE tf-%d real %s' % (i, words_to_check[i]))

# Class always must go last
out.append('@ATTRIBUTE %s {%s}' % ('class',
', '.join(['suicide', 'non-suicide'])))

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

```

```

features = {}

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

[clang, perc] = identifyLang(checkText)

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

features['perc'] = str(perc)
features['clang'] = str(clang)

row = []

        features['length'] =
str(offline_parser.get_feature_length(checkText
, lang = clang))
        row.append(features['length'])

        features['number_of_sentences'] =
str(offline_parser.get_feature_number_of_sentences(checkText.encode('utf8'), lang = clang))
        row.append(features['number_of_sentences'])

        features['word_count'] =
str(offline_parser.get_feature_word_count(checkText, lang = clang))
        row.append(features['word_count'])

        features['words_per_sentence'] =
str(offline_parser.get_feature_words_per_sentence(checkText.encode('utf8'), lang = clang))
        row.append(features['words_per_sentence'])

        features['text_shannon_entropy'] =
str(offline_parser.get_feature_text_shannon_entropy(checkText, lang = clang))

row.append(features['text_shannon_entropy'])

for attr in read_scores:
    call = getattr(offline_parser.textstat,
attr)

    features[attr] = call(checkText)
    row.append(str(features[attr]))

# For sentiment we need english
        features['in_english'] =
offline_parser.get_translated_data(checkText,
clang).strip()

        features['sentiment'] =
offline_parser.get_feature_sentiment_score(features['in_english'])
        row.append(features['sentiment'])

        features['misspelling'] =
offline_parser.get_feature_mispelling_score(checkText, lang = clang)
        row.append(features['misspelling'])

# TFIDF
tf_dump =

```



```

pickle.load(open('suicide_v1c_withtfidf_30_ngram1.arff.tf.safe.pickle', 'rb'))

# Ekatharisi tou text mono me tis lekseis pou thelw
temptext = features['in_english']
temptext_out = []
temptext_stats = {}
temptext_all_words = []
for word in temptext.split(' '):
    word = word.lower()
    word = re.sub(r'([^\a-z ])', '', word)
    temptext_all_words.append(word)
    if word != '' and word in words_to_check:
        temptext_out.append(word)
        temptext_stats[word] = temptext_stats.get(word, 0) + 1

features['in_english_tfidf_stripped'] = '.join(temptext_out)
for i in range(len(words_to_check)):
    features['word_%s' % words_to_check[i]] = temptext_stats.get(words_to_check[i], 0)
    if len(temptext_stats) > 0:
        row.append('%f' % (temptext_stats.get(words_to_check[i], 0) / len(temptext_all_words)))
    else:
        row.append('%f' % 0)

# Tag to search
row.append('?')

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

f.close()

return features

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

    features = {}
    various_info = {}

    create_frailsafe_arff(oldText, newText)

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

    os.unlink(temp_file_out)

    various_info['temp_file_out'] =

```

```

temp_file_out
    various_info['command'] = command
    various_info['raw_res'] = res

    ret = {}
    ret['oldText'] = oldText
    ret['newText'] = newText
    ret['prediction'] = 'na'

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

    ret['features'] = features
    ret['various_info'] = various_info

    return json.dumps(ret)

@app.route("/predict_suicidal", methods=['POST'])
def predict_suicidal():
    checkText = request.form['checkText']

    various_info = {}
    features = create_suicidal_arff(checkText)

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

    os.unlink(temp_file_out)

    ret = {}
    ret['checkText'] = checkText
    ret['prediction'] = 'na'

    various_info['temp_file_out'] = temp_file_out
    various_info['command'] = command
    various_info['raw_res'] = res

    temp = res.replace(':', '').replace('-', '_').strip().split(' ')
    if len(temp) > 5:
        ret['prediction'] = temp[5]
        ret[temp[6]] = temp[7]
        features[temp[6]] = temp[7]

    ret['features'] = features
    ret['various_info'] = various_info
    return json.dumps(ret)

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

```

## 9.4 Predictor

File: predictor/predictor-cli.java

```

1. package predictor;
2.
3. import weka.classifiers.Classifier;
4. import weka.core.Instances;
5. import weka.core.converters.ConverterUtils.DataSource;
6.
7. public class PredictorCLI {
8.
9.     public static void main(String[] args) {
10.
11.         Classifier cls;
12.         try {
13.             //Load model
14.             cls = (Classifier) weka.core.SerializationHelper.read("frailsafe.model");
15.
16.
17.             DataSource source;
18.             try {
19.                 //Load test data
20.                 source = new DataSource("in.arff");
21.                 Instances data = source.getDataSet();
22.                 if (data.classIndex() == -1)
23.                     data.setClassIndex(1); //class attribute is the second attribute
24.
25.                 //predict & print
26.                 for(int i=0; i<data.numInstances();i++){
27.                     double value=cls.classifyInstance(data.instance(i));
28.                     String prediction=data.classAttribute().value((int)value);
29.                     System.out.println("Prediction for instance: "+i+" is: "+prediction);
30.                 }
31.
32.             } catch (Exception e) {
33.                 // TODO Auto-generated catch block
34.                 e.printStackTrace();
35.             }
36.         } catch (Exception e) {
37.             // TODO Auto-generated catch block
38.             e.printStackTrace();
39.         }
40.     }
41.
42. }
```



## 9.5 FrailSafe demo video

This image is also in Annex 10.2 of **D4.13** as both reports are responsible for this demo.

File: *frailsafe-showcase.wmv* (*frailsafe-showcase.jpeg*)

File: *frailsafe-showcase.wmv*

Size: 22326880 bytes (21,29 MiB), duration: 00:04:00, avg.bitrate: 744 kb/s

Audio: wmv2, 48000 Hz, 2 channels, s16, 192 kb/s (eng)

Video: wmv3, yuv420p, 1280x720, 5000 kb/s, 25,00 fps(r) (eng)

FrailSafe Demo video

