Software Requirements Specification (SRS) Document

Hello*String* Application

30/9/2022

Version 2.4

Alexandros I. Karasakalidis

Table of Contents

1. Introduction

1.1 Purpose

This document contains a detailed description of the Hello*String* application. It explains the application's features, it's interfaces and it's constraints.

1.2 Intended audience

This document is intended for the main stakeholder of the project, Dr. G. Tsoulouchas, as well as the developer of this project.

1.3 Scope: Specify how the software goals align with the overall business goals and outline the benefits of the project to business.

Hello*String* application will provide the ability to it's end users to use common computational linguistic tools to analyse a given text, without requiring any expertise on the subject. Features like part-of-speech tagging and sentiment analysis tools will be available to the end user.

The tools will be split into two sets - one set will be free to use from every user, while the other will be available only for registered and verified users.

A registered user will also be able to access a short history of his interactions.

2. General Description

2.1 Product features

A front-end interface in the form of a webpage will be the main interface of the end user with the application. Nonetheless, an API will also be provided with more customization and control over the tools as well as for giving the ability to more experienced users to utilize the API to automate text processing.

The tools provided with this application to the end user are split into two sets. The first set includes tokenization, word/character frequency analysis, stopword removal, stemming, lemmatization and POS tagging tools. The second set includes sentiment analysis and aggressiveness scoring tools.

The first set of tools will be available to all end users while the second set of tools will be available only to registered and verified users. This split takes place because of the significantly larger computational needs of the second set over the first. With that in mind, in order to prevent an exploitation of our resources, we require that users who need to use these tools to be verified.

A simple user database will be included in the application. A user that registers will need to verify his email with a verification email that he will receive.

Finally, a verified user will be able to review his last transactions with the application, either that happened through the front-end or through the API. A privileged user however will be able to query all of the transaction history of each user.

2.2 User class and characteristics:

This application will consider three types of users:

- End-User: May use the interface or the API but is limited to the first set of tools.

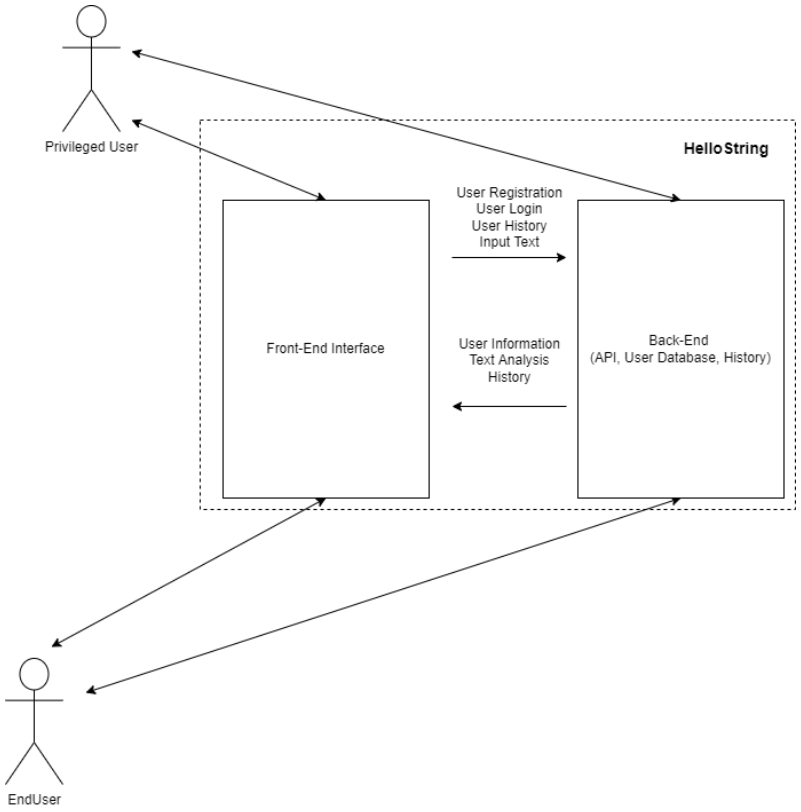
- Verified End-User: May use the interface or the API and the full set of tools included with this application.

- Privileged User: May access the history of transactions of each verified end-user.

The End-User could be characterized as expected to be at least Internet literate, in order to access the front-end interface. In order to use the API, the End-User is expected to be familiar with RESTful API interactions.

The Privileged User is also expected to be at least Internet literate to exercise his ability either through the front-end interface or – for an advanced user - through the RESTful API.

2.3 Operating environment



Both End-Users and Privileged users access the system either through the Front-End or the Back-End application.

2.4 Constraints

The front-end interface limits the input text's length to 100 characters. This limitation does not apply to the API. This constraint is because the sentiment and aggressiveness scoring models are performing reasonably well when the given text is small in length.

Unregistered end-users can not be monitored as we can have no way of relating history logs with them.

2.5 Assumptions and dependencies

User management is non-existent, as it is considered out-of-scope for this application. Furthermore, the front-end and the back-end will are built as two separate systems. The back-end will be able to operate without the front-end.

While the back-end can be isolated from outside connections, the front-end uses components that require an open connection to the Internet.

3. System Requirements

3.1 Functional requirements

We will state the functional requirements through user use cases:

End User: Register

- Description**:**

  The user may use the front-end interface to create an account by having access to a registration page after he enters the required information the user is now a registered user and he can log in at any time.

End User: Delete User

- Description**:**

  The user may use the front-end interface to delete his account at any time. A user settings page will exist that will allow the user to delete his account at any time.

End User: Verify User

- Description**:**

  After registration, an email will be sent to the email address of the registered user. After following a link into that email, the user will be verified from that moment. A user will have the ability to resend that verification email through his e-mail.

End User: Analyse Text

- Description**:**

The user may use the front-end interface to analyse any desired text. He will have a page where he can input his text and once submitted, the system will present the user with the analysis done on his text. The interface may provide the user a part-of-speech tagging analysis, a frequency analysis as well as sentiment/aggressiveness analysis if the end-user is verified.

End User: API Documentation

- Description**:**

  The user may use the front-end interface to get access to the API documentation regardless of if he's registered or not. A page should be included with a detailed description of every method offered by the API.

End User: API

- Description**:**

  The user may use freely the API methods provided by this application regarding the first set of tools (see 2.2 Product features). These API methods should be adequately described on the API documentation available to those users.

Verified End User: History

- Description**:**

  The user may have access to his last interactions with the application. This short history will be available on the homepage of the front-end interface.

Verified End User: Authentication Key

- Description**:**

  The user may use the front-end interface to access his authentication key. The authentication key will be available at the user settings page for every verified user.

Verified End User: API

- Description**:**

  The user may use freely any API methods. These API methods should be adequately described on the API documentation available to those users.

Privileged User: Monitor

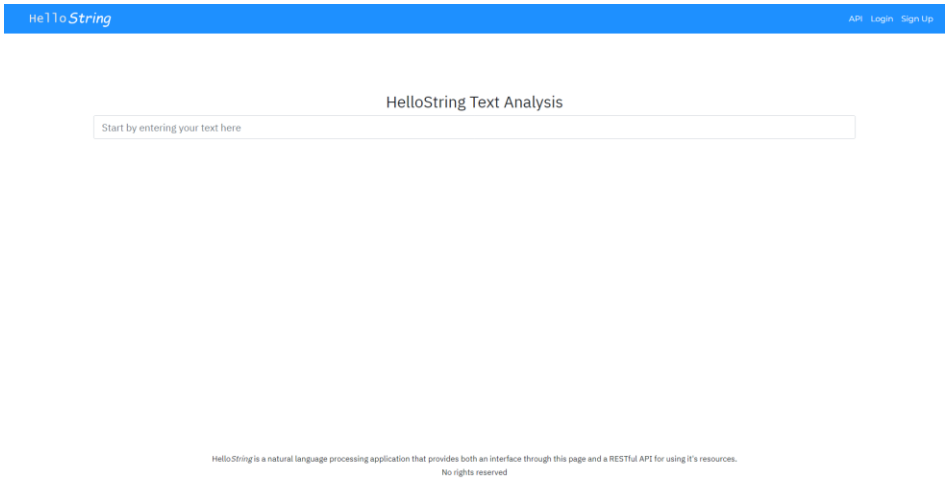
- Description**:**

The user may monitor the transactions of each verified user. A page will be provided to him through the front-end interface, where he will be able to query the transactions of a selected user.

4.External Interface Requirements

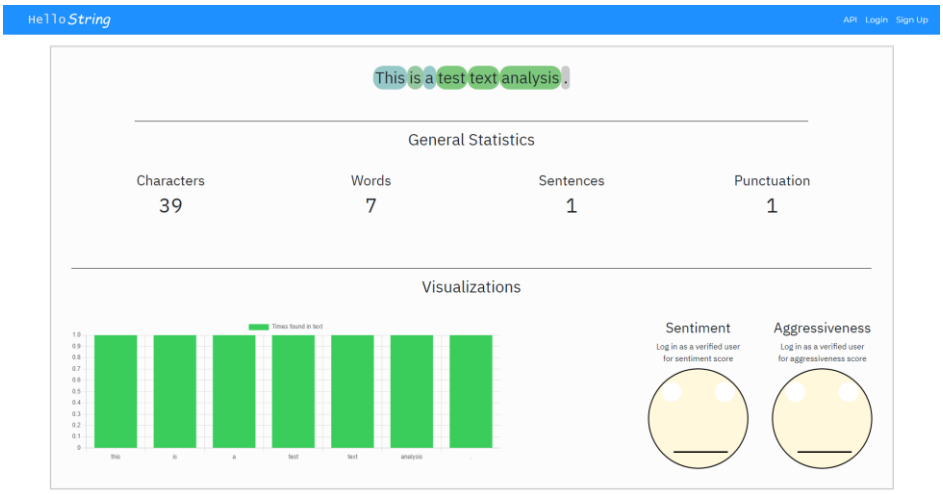
4.1 User Interfaces

The front-end homepage:



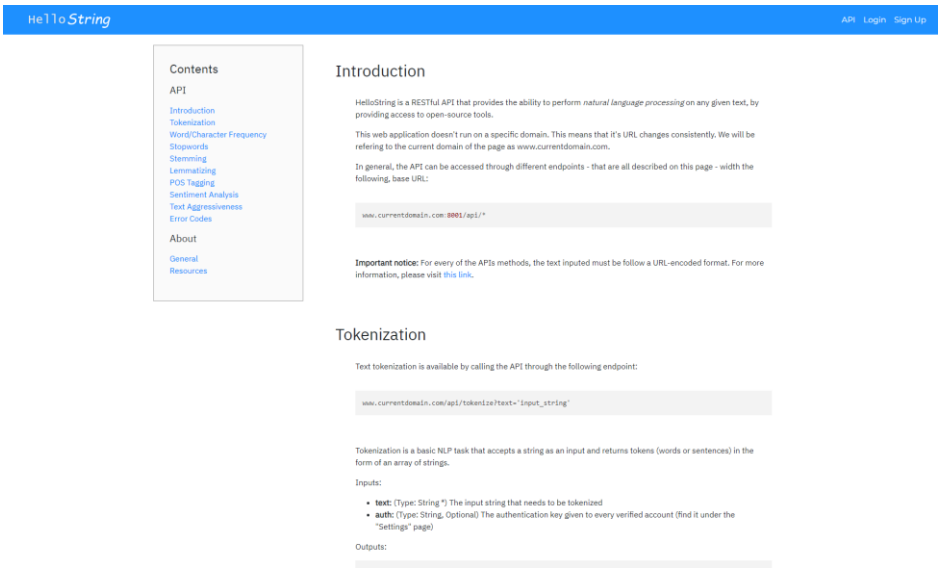
The user may enter a string for analysis, or he may use the navbar on the top of the page to navigate to the API Documentation page, the Login page or the registration page (Sign Up).

The front-end text-analysis page:



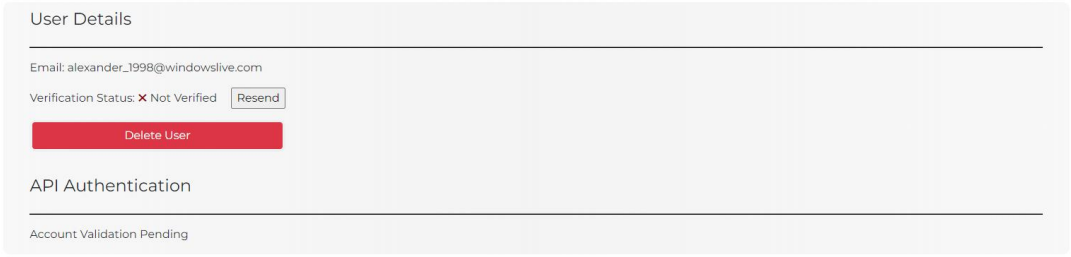
The user is presented the analysis set up for the front-end interface. Limited interactivity is available through the text's words or the frequency graph.

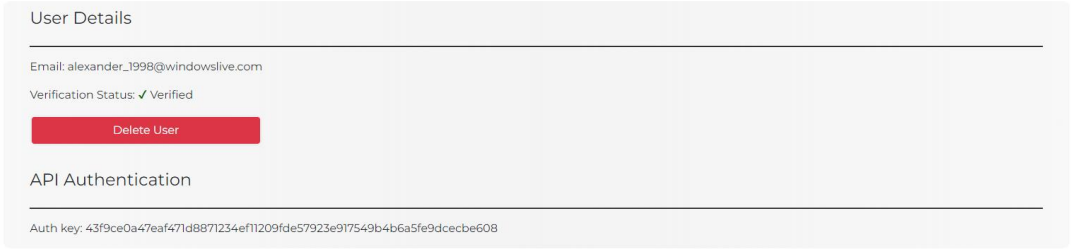
The front-end API Documentation page:



The main source of information about the API methods for a verified - or not – user.

The Settings page:





The settings page where the user may delete his account, resend his verification email or – if he's verified – have access to his authentication key.

4.2 Hardware/Software Interfaces

The front-end interface is built for desktop use (Windows, Linux or Mac OS environments) with any modern web browser installed. Smartphones can access the application but are not supported officially.

Directly calling the application's API has no limitations in terms of interface.

5. Non-Functional Requirements

5.1 Logical Structure of Data

User Table

Name	Type	Description
id	Primary Key (Integer)	Primary key of User Table
email	String – Unique	Email of the user account. Must be unique.
password	String	User's account password. Is stored as a hash.
auth_key	String	User's authentication key. Is stored as a hash.
mail_auth_key	String	User's verification code. Is used to verify user. Is stored as a hash.
is_authed	Boolean	Is the user authenticated?
is_privileged	Boolean	Is the user privileged?

Text_Log Table

Name	Type	Description
id	Primary Key (Integer)	Primary key of Text_Log Table
auth_key	String	User's authentication key. Is used to link Text_Log record with a user record.
parameters	String	Parameters of the transaction – applicable if transaction describes direct API call.
text	String	Inputted text.
rest	Boolean	Is the transaction done through a direct API call?

Logical structure of the Data:

5.2 Security

Every password and authentication code is stored as a hash. Furthermore, user verification requires mail verification.

User history exposure is only available through privileged user's authorization keys.

A user may become privileged with the usage of an API method available for usage only by a unique authentication key holder (available only to system administrator/developer).

A recommendation - not applied as of the release of the application – would be for an API call interval period, so that brute-force attacks on authorization keys would be severed.