SE 3XA3: Software Requirements Specification: Rev 1
Plagiarism Check

Team 310, MXQ Gang
Qifeng Xu, xuq14
Xu Wang, wangx147
Mrinal Tiwari, tiwarim

April 7, 2020

Contents

List of Tables

List of Figures

Table 1: **Revision History**

Date	Version	Notes
Feb 9, 2020	1.0	added the requirement specification
April 6, 2020	1.1	updated the requirement specification

1 Project Drivers

1.1 The Purpose of the Project

This project is to develop a plagiarism checker that can detect the similarity ratio of two documents. With the development of the internet, more and more documents and information can be accessed and referenced. Therefore, the people who cite a lot of documents need to make sure their work is unique. The purpose of this software is to avoid plagiarism for writing essays and documentations. It is able to assist students and researchers to accomplish better documents.

1.2 The Stakeholders

1.2.1 The Client

The client of this software is the people and institutions that require the service of plagiarism checking continuously. They are the people and institutions that need this service in their daily work such as researchers and universities.

1.2.2 The Customers

The customer of this software is the general public whoever needs to check for plagiarism for their documents. The majority of the customer for this product will be student who is working on a project or essay.

1.2.3 Other Stakeholders

The people who are interested in this service can also be our stakeholders. They may only need to use this product once or twice, but these people can be major users of this software.

1.3 Mandated Constraints

1.3.1 Solution Constraints

Constraint: This software shall operate correctly on devices (e.g. Desktop, Laptop, cell phone and tablet) with a browser that is compatible with python.
Rationale: The clients use correct browsers to execute this software.

Fit Criterion: The software shall be approved to operated correctly if the users are using appropriate browsers.

1.3.2 Schedule Constraints

This project shall be completely finished by April 6, 2020.

1.3.3 Budget Constraints

The purposing budget for this project is 0. Since this project is focusing on redeveloping and improving an existing software, it does not require any budgets on it.

1.4 Naming Conventions and Terminology

Table 2: **Table of Terms and Abbreviations**

Term	Definition
REST	method of using API over internet
API	routines for building software tools
Containers	standalone images and libraries needed
Docker	containerization tool
Kubernetes	orchestration tool
Orchestration Tool	tool used for scaling the container
~~React~~	~~C# framework for making UI~~
Python3	programming language used to implement the functionality
HTML 5	programming language to make a static website
CSS 5	language used to give styling to a static website
JS	JavaScript programming language used to make a website interactive
GUI	Graphical User Interface

1.5 Relevant Facts and Assumptions

1.5.1 Relevant Facts

The existing software is 264 lines of python code. This software does not have a license since it was previously developed by one of ~~our group members~~ the team member on the MXQ Software Development team of ~~for~~ this project.

1.5.2 Assumptions

It is assumed that the users are able to connect to internet when they need to use this software and they can understand the basic operation of this software.

2 Functional Requirements

2.1 The Scope of the Work and the Product

2.1.1 The Context of the Work

Plagiarism Check is a further development to the open source Plagiarism Check REST Api which checks similarity between two documents and gives a hit for plagiarism. Further development to this project involves modifying existing code for better readability and maintainability, adding a GUI to the project in the form of an interactive website allowing registration of user, detection for similarity and refilling of tokens to access the features.

- Requirements Document Revision 0 February 9
- Proof of Concept Demonstration February 10
- Test Plan Revision 0 February 28
- Design Documentation March 13
- Demonstration Revision 0 March 16
- Final Demonstration Revision 1 March 30
- Final Documentation Revision 1 April 6

2.1.2 Work Partitioning

Table 3: Work Partitioning Part I

Event Number	Event Name	Input	Output
1	Plagiarism Check Creation	Developer code	Internet Browser
2	Text Input Reading	Text	
3	Plagiarism Check	Text	Result
4	Register account	Text	Result
5	Refill tokens	Text	Result

Table 4: **Work Partitioning Part II**

Event Number	Summary of BUC
1	Create the Internet browser based on the code
2	Record the text that user has input
3	Start comparing the two text and return the result
4	Finish edits to the project

2.1.3 Individual Product Use Cases

2.2 Functional Requirements

- **FR1:** The website shall allow user registration using username and password.
  ~~Fit Criterion / test case: Gives a status code of 200 when user hits Register button.~~
  Rationale: To allow a personal account for every user using the Plagiarism Check service.

- **FR2:** The website shall allow user to login using unique username and password.
  ~~Fit Criterion / test case: There should not be multiple users in database~~

~~with the same username. When trying to register same username twice, website should display an error.~~
Rationale: There should not be two users with the same username.

- **FR3:** User's password shall not be stored in database as the original password.
~~Fit Criterion / test case: The password stored in the database and the original password should not match. The password in database should been hashed and should look unlike natural language.~~
Rationale: For security and safety from malicious hacks, the user's credentials should not be stored in true form.

- **FR4:** User shall be able to enter the two texts it wants to compare for plagiarism.
~~Fit Criterion / test case: Check the webpage, it should contain two text boxes to enter each text1 and text2.~~
Rationale: A way of getting inputs from the user.

- **FR5:** Web-page shall have a submit button to submit the two texts to compare.
~~Fit Criterion / test case: Check the webpage if it consists a button specified for submitting inputs and calling the API.~~
Rationale: A method to send inputs and accept output.

- **FR6:** The website shall allow a user to refill his tokens whenever required.
~~Fit Criterion / test case: After refilling tokens, the present token should be sum of token refill amount and original number of tokens.~~
Rationale: A way to monetize the service by using tokens and getting paid to increase tokens for users.

- **FR7:** The API shall return status code for each API call.
~~Fit Criterion / test case: In the terminal, there should be status code displayed for each API call.~~
Rationale: A way to tell if the backend is working as expected.

3 Non-functional Requirements

3.1 Look and Feel Requirements

3.1.1 Appearance Requirements

The product should display two text boxes and some buttons on the website. It would choose a suitable color as its background color. Also, there would be some instructions to help the user understand what to do next. The product should display a numerical result after user has submit the text.

3.1.2 Style Requirements

Plagiarism Check would appear as a literal website. It should make the user feel comfortable when they use the product.

3.2 Usability and Humanity Requirements

Ease of Use Requirements

The product should be easy to use for any individual who is studying or working in high school and universities.

3.2.1 Personalization and Internationalization Requirements

All text displayed by interface shall in English with Canadian spelling.

3.2.2 Learning Requirements

The user shall be able to operate an internet browser and a mouse.

3.2.3 Understandability and Politeness Requirements

The user shall be able to operate an internet browser and a mouse.

3.2.4 Accessibility Requirements

The product shall be usable by partially sighted users.

3.3 Performance Requirements

Speed and Latency Requirements

The result shall be displayed immediately after user submits the text.

3.3.1 Safety-Critical Requirements

The product shall not compromise the user data or internet environment.

3.3.2 Precision or Accuracy Requirements

The product shall use integer numbers outside the text input area.

3.3.3 Reliability and Availability Requirements

The product shall be available for use 24 hours per day, 365 days per year wherever there is an internet connection.

3.3.4 Robustness or Fault-Tolerance Requirements

The product would not save text if website crashes, but there would be a message explaining that the program unexpectedly quit when relaunched.

3.3.5 Capacity Requirements

The game shall not exceed the server load.

3.3.6 Scalability or Extensibility Requirements

The code will allow for scalability.

3.3.7 Longevity Requirements

The product shall be relevant to the lifetime of JavaScript functional browsers and website server.

3.4 Operational and Environmental Requirements

Expected Physical Environment

The product shall be used in any environment that contains access to the internet.

3.4.1 Requirements for Interfacing with Adjacent Systems

The product shall be able to run on many different devices as long as the devices can open web browsers.

3.4.2 Productization Requirements

The product shall be hosted by a website for local use before it is sold.

3.4.3 Release Requirements

The program shall update yearly and satisfy the customer's demand.

3.5 Maintainability and Support Requirements

Maintenance Requirements

Make maintenance of the product minimal.

3.5.1 Supportability Requirements

Ensure the potential users could run the program on their local machines.

3.5.2 Adaptability Requirements

The game shall be universally acessable by users.

3.6 Security Requirements

Access Requirements

All the public visitor shall be able to access the product on the website.

3.6.1 Integrity Requirements

The product shall not alter its source code.
The product shall prevent incorrect data input.

3.6.2 Privacy Requirements

The product shall not divulge data submitted or any user's personal information.

3.6.3 Audit Requirements

N/A

3.6.4 Immunity Requirements

N/A

3.7 Cultural and Political Requirements

Cultural Requirements

The product shall not be offensive to religious or ethnic groups. The product shall be available in English.

3.7.1 Political Requirements

N/A

3.8 Legal Requirements

3.8.1 Compliance Requirements

The product shall not compromise any laws.

3.8.2 Standards Requirements

N/A

3.9 Health and Safety Requirements

N/A

4 Project Issues

4.1 Open Issues

N/A

4.2 Off-the-Shelf Solutions

Ready made products: Grammarly

Reusable code components: Natural language processing model Spacy

Products that can be copied: This being a re-development of an open-source project, the original implementation, "Plagiarism Check REST API" can be used as a source code.

4.3 New Problems

Effects on Current environment: Our dynamic website would be one of the hundreds of millions of website that get hosted on the server. However, if the server gets overloaded by significantly larger number of users at the same time, it could result in server crashing and could take down all the websites on the server including ours.

Effects on the installed system: This project of ours is implemented in a containerized manner using Docker and thus each container is standalone and independent. Crash in one container would not affect other containers.

Potential User Problems:
N/A

Limitations in the Anticipated Implementation Environment that may inhibit the new product: The website might have a bit more latency than desired due to use of trivial HTML and JS. If we encounter such

Table 5: **Task time-table**

Deliverable, schedule	
Deliverable name	Due by
Problem Statement	January 24, 2020
Development Plan	January 31, 2020
HTML page for login	February 5, 2020
HTML page for user input and similarity result	February 7, 2020
Requirement Document Revision 0	February 9, 2020
Proof of Concept Demo	February 10, 2020
HTML page ready for refilling tokens	February 18, 2020
Test Plan Revision 0	February 28, 2020
Linking HTML to REST Api, testing	Marc 10, 2020
Design and Document Revision 0	March 13, 2020
Revision 0 Demo	week of March 16
Final Demo Revision 1	week of march 30
Final Demo	week of march 30
Final Documentation Revision 1	April 6, 2020

latency, we could overcome it by using React for front-end.
There is also a possibility of high user traffic which could result in server crash or slow performance of our website. We could use orchestration tools like Kubernetes to scale our containers automatically in horizontal or vertical direction as required to handle more user traffic.

Follow-up problems: It would be difficult to cope with the problem if any of the developers in the team quits as each of the developers in the team has its own strength and weakness and the work is divided based on the strengths.
There is also a follow up problem of web browsers releasing new updates every once in a while or a whole new browser getting released making our code implementation void or non-functional.

4.4 Tasks

Our project's task is set according to the deliverable set for Software Engineering 3XA3 course-work and team milestones. Refer to Table 2.

4.5 Migration to the New Product

Requirements for migration to the New Product: A dynamic interactive website that lets user login and enter texts and get the similarity ratio.

Data that has to be modified or translated for the new system: none

4.6 Risks

There is a risk of low quality of the website and low productivity of the development team during to unforeseen circumstances in future.

4.7 Costs

There are no costs involved at present but this could change if any of the currently used technologies or web hosting service starts to monetize its service in future.

4.8 User Documentation and Training

~~N/A~~

To use Plagiarism Check please visit "http://plagiarismcheck-com.stackstaging.com". To register for this service, use Register section of the website to fill in the desired username and password and click on "Register" button.

Once the user is registered, use Detect section of the webpage to enter two texts to compare and click on "Submit" button. In a moment, a table would appear containing the similarity ratio between the two texts, message and number of tokens left for the user.

User can also refill his tokens by giving inputs Admiral password "Admiral123" and amount to refill and clicking "Submit" button in the Refill section.

4.9 Waiting Room

- Implementation of front-end in React for a better UI.

- Support of complex data like charts and images.
- Hosted using Kubernetes clustering for scaling.

4.10 Ideas for Solutions

Token shift of credentials when user goes from one URL to another in order to have authentication.