Software Requirements Specification

for

NER Text Annotation Manager

Clients
Torsten Hahmann
Umayer Reza

SEWDO Developers
Sam Waggoner
Wilder Baldwin
Darien Orethun
Owen Bellew

sewdo

sewdo

NER Text Annotation Manager
System Requirements Specification Document

Table of Contents

1. Introduction 2
    1.1 Document Purpose 2
    1.2 External References 2
    1.3 Product Motivation and Purpose 3
    1.4 Product Scope 3
2. Functional Requirements 6
    2.1 Use Cases 6
    2.2 Functional Requirement Testing 12
3. Non-Functional Requirements 18
    3.1 Summary of Non-Functional Requirements 18
    3.1 List of Non-Functional Requirements 19
4. User Interface 21
5. Deliverables 23
6. Open Issues 24
Appendix A – Agreement Between Customer and Contractor 24
Appendix B – Team Review Sign-off 25
Appendix C – Document Contributions 25

Name	**Date**	**Reason for Changes**	**Version**
All	10/22/23	Original creation	1.0
All	10/30/23	Incorporating peer feedback, reworking FRs & tests	1.1

1. Introduction

The NER Text Annotation Manager is an existing software tool that allows users to add annotations of text, which can then be used for Named Entity Recognition (NER) tasks. Text annotations are labels of words or sections of a sentence. For example, 'sulfuric acid' and 'chlorine' could both be labeled as 'chemical', which would help the NER model learn what chemicals are. This project aims to add or improve three primary features: annotation review, annotation editing, and loading and exporting provenance data. The project aims to include these features with a user interface that enables quick, simple, and intuitive annotation reviews. The addition of these features is the capstone project for the members of our team (Sam Waggoner, Wilder Baldwin, Darien Orethun and Owen Bellew), in partial fulfillment of the Computer Science BS degree for the University of Maine. This project was requested by Torsten Hahmann and Umayer Reza.

1.1 Document Purpose

This document is designed to overview the requirements and specifications of the project. It is intended to be addressed to the stakeholders of the project, the client, and the users. It contains an overview of the motivation and goals of the project as well as a list of technical specifications of the program and the project deliverables. Lastly, this document has a living section at the bottom of contact between the developers and the clients.

1.2 External References

This section includes citations for the external references that were mentioned or utilized in this document. It is presented in two ways for the reader's ease of accessing links and information.

NER annotator for SpaCy
Owner: GitHub user Tecoholic
Date: Created 8 November 2020
Web App URL: https://tecoholic.github.io/ner-annotator/
GitHub URL: https://github.com/tecoholic/ner-annotator

UML Distilled
Author: Martin Fowler
Date: Published 2003
URL: https://martinfowler.com/books/uml.html

List of Chromium Releases
Source: GitClear
Date: Current
URL: https://www.gitclear.com/open_repos/chromium/chromium/releases

Bibliography

Arunmozhi. (2023). *NER Annotator for Spacy* [Vue]. https://github.com/tecoholic/ner-annotator (Original work published 2020)

Chromium Releases—GitClear. (n.d.). Retrieved October 30, 2023, from https://www.gitclear.com/open_repos/chromium/chromium/releases

Fowler, M. (2003). *UML Distilled*. https://martinfowler.com/books/uml.html

1.3 Product Motivation and Purpose

Umayer Reza, a graduate student at the University of Maine, is working with faculty professor Torsten Hahmann. Reza's Ph.D project involves creating a knowledge ontology in a domain within forestry and agriculture. His ultimate goal is to have a tree of sorts, with nodes being states and edges being actions. For example, by looking at this tree one could see that it is possible for some substance to go from a solid state to a liquid state by heating it at 400 degrees F. To create this tree, he is using a language model that reviews published literature on the subject of forestry and agriculture, which is then able to learn the states, and the relationship between them. However, the raw sentences in the literature are not enough to train this model–to learn the content, it needs labels indicating what words are the states and actions. To get these labels, experts in the field must manually label the text, sentence by sentence. However, the tool that they use to create these labels is time-consuming to use in several key ways. This is where Umayer Reza and Dr. Hahmann requested the help of SEWDO. Our project is to allow for the extension of the base SpaCy-en NER model by hand annotators. Our goal is to add editing and reviewing features to existing annotations, as well as adding the ability to load and export the provenance data in JSON format. The project will have an emphasis on an intuitive and user-friendly interface. By adding these features, human annotators will be able to produce labels much faster and much easier, thus creating more data and improving the model that Reza is creating. In addition, the field experts' time is valuable. These improvements will mean that it will be easier to train new annotators and contributions will demand less of experts' time.

1.4 Product Scope

This section includes the essential use case diagrams that describe how the product will be used. Editable diagrams can be seen at this link. These diagrams follow UML practices similarly to as they are in *UML Distilled* (Fowler, 2003). To explain it verbally, there are two primary use cases of an annotator. The first use case is if they want to create new annotations on a text document. This is referred to as Annotation Mode.

In Annotation Mode, users can upload documents (either text documents or as previously annotated text documents formatted as JSON). They can then edit annotations. These edits can include adding a label, deleting a label, or replacing a label with a different label. After they are done editing, they can enter their name then export their annotations. This export file will be formatted as a JSON, and it will include the history of the changes that the user made, including their name, the date, and the time of all of the changes they made. This historical tracking will allow people to see who made what changes, and when.

Users would use Review Mode if they want to go over annotations that others have made, instead of creating them from scratch themselves. The primary use case of this mode is reviewing annotations created by a named-entity recognition (NER) model. After the model generates its own annotations for a piece of text, a human expert in the field is needed to confirm that the labels are correct, before they can be used to generate an ontology (or before they are used for any other purpose). To accommodate for this situation, this mode allows the annotator to accept, reject, or suggest a different label for every annotation for every annotation in the document. Once they are done, just like in Annotation Mode, they can export their new changes with the history included in the JSON.

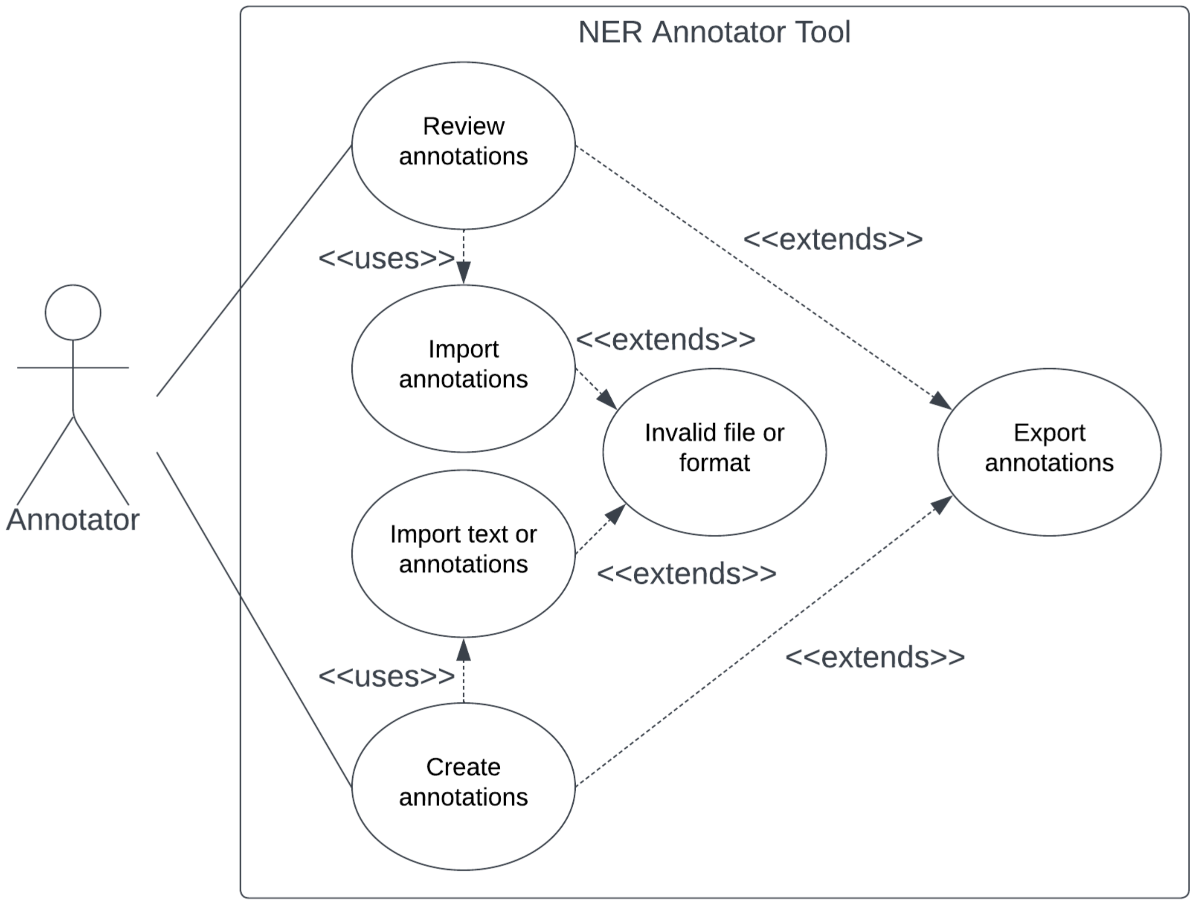


Fig. 1.4.1: Top level use diagram

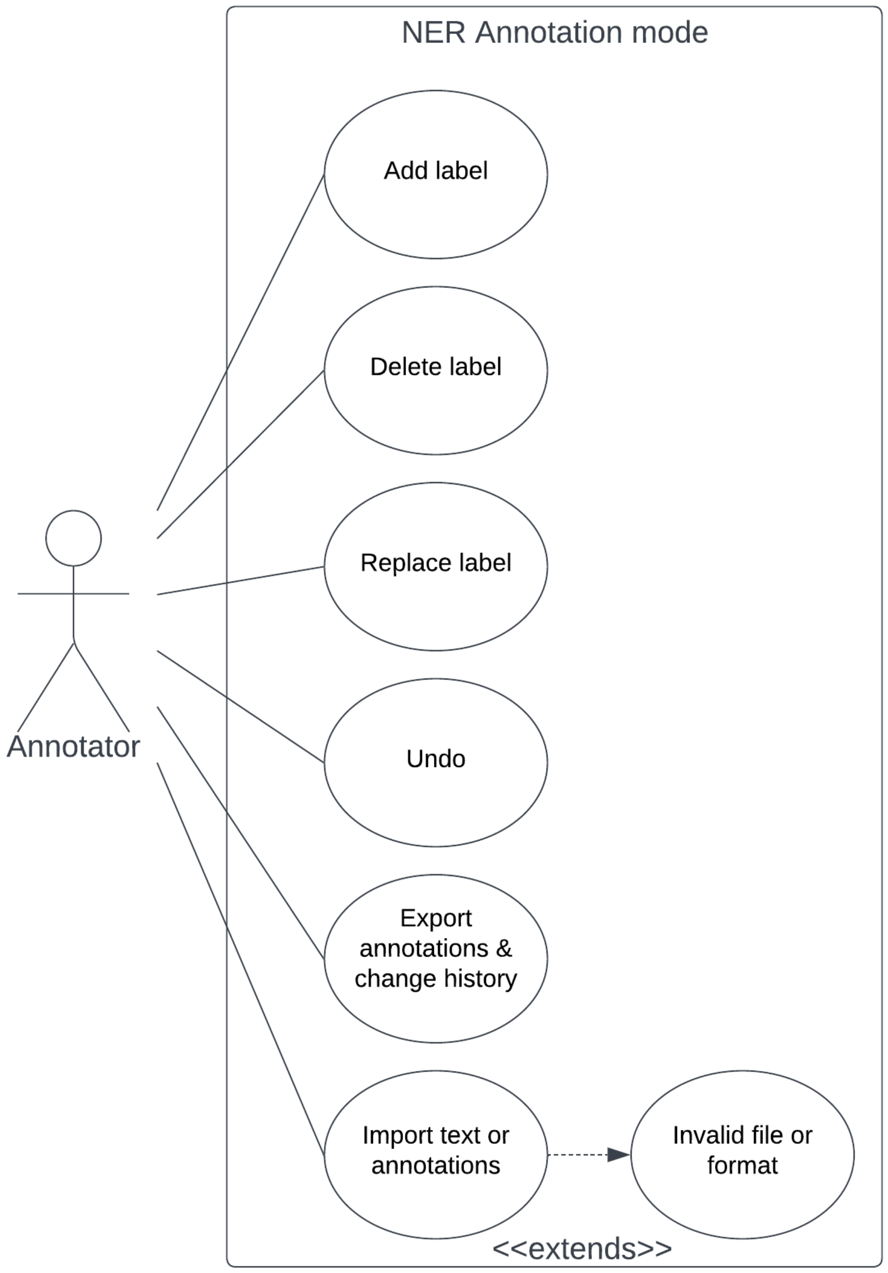


Fig. 1.4.2: Annotation mode use case diagram

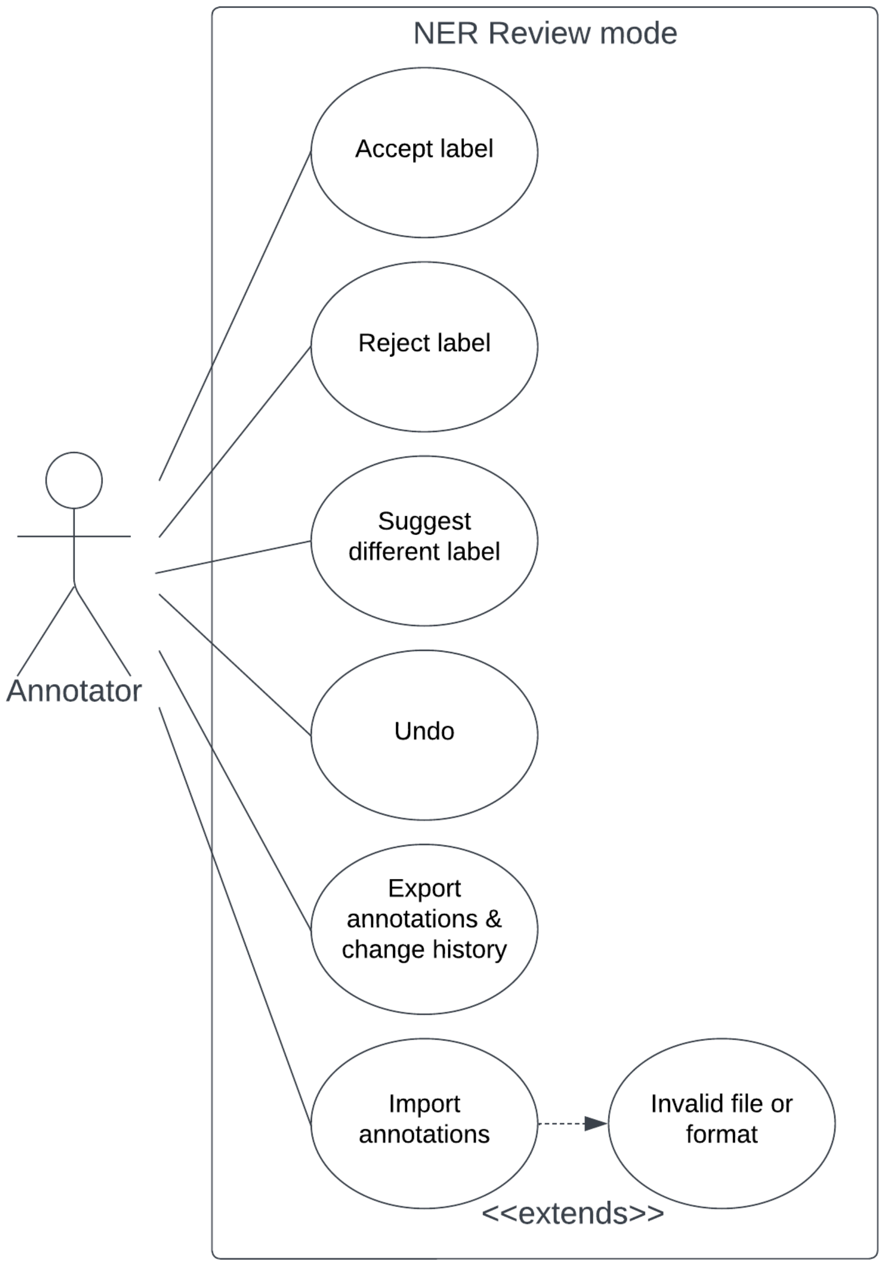


Fig. 1.4.3: Annotation mode use case diagram

2. Functional Requirements

2.1 Use Cases

This section includes all of the functional requirements (presented as use cases) for the application. Primary users for this application are expert annotators. The most applicable usage is by faculty from the forestry department at the University of Maine. Since the project builds on an existing application, several key functions are already implemented. For example, text import is already implemented in the current application, but JSON annotation import is not. So although both are included in the use case diagram, import text is not listed in the requirements. Directly below is a list of changes that will be made, in context of the mode with which they are associated.

- Annotation and Review Mode
    - modify functionality: export single json (instead of 3: text, annotations, and tags), including change history (change's date, time, author)
    - modify functionality: import json annotations
        - When you upload a file, the system should detect if it is raw text or annotations, and include the annotations if they are present
    - add functionality: undo actions
        - For annotation mode: adding, replacing, deleting annotations
        - For review mode: accepting, rejecting, suggesting different labels, reopening labels
- Annotation Mode
    - add functionality: replace label with another label
- Review Mode
    - add functionality: create this mode
      (in this mode, you can't add more annotations, you can only accept label, reject label, suggest replacing labels, and Re-Open labels)
    - add functionality: Accept label, reject label, suggest different label (replacing delete label)
    - add functionality: reopen review

ID	FR-1
Name	Import JSON Annotations
Summary	In either mode, the user imports a single JSON file of annotations. The system will be able to accommodate both text (.txt) files as well as annotated JSON files.
Priority	5
Preconditions	User has a single JSON file that includes the text as well as the associated annotation labels, tags, and history of changes of each label (including date, time, and author). It could be the case that there are no associated labels, etc. and it is just text.

Postconditions	Program will have loaded the text along with the associated annotation tags and tag colors, if they exist. If the file is exported once again, the previous history will be included in the new export JSON.	
Primary Actor	Expert Annotator	
Secondary Actors	None	
Trigger	Two possible triggers: - Clicking “Open a text file to begin” or dragging file onto that field - Clicking File > Review Annotation File	
Main Scenario	**Step**	**Action**
	1	User clicks ‘Import’ under ‘Review annotations’
	2	System opens file explorer
	3	User selects file to import
	4	System loads the text file and displays the editing screen with all existing annotations
Extensions	**Step**	**Branching Action**
	1a	User drags annotation file onto the ‘Import’ box under Review Annotations - The same process will be followed. Skip to step 4
	4a	Incorrect file type, format, or unreadable content**:** - System creates error message, goes back to previous screen before upload was clicked
Open Issues	None	
References	Fig. 1.4.1, Fig. 1.4.2	
Related Use Cases	All use cases. This use case is the precursor to every single other use case. However, since the current functionality already allows you to upload a text document then make annotations, the same outcome can be achieved with the current application.	

ID	FR-2	
Name	Export JSON Annotations	
Summary	The user exports a single JSON file of annotations.	
Priority	5	
Preconditions	In either annotation mode or review mode, the user has imported text or an annotated JSON and optionally made annotations or reviews.	
Postconditions	The file is saved in downloads, with a name incorporating the date on which the file was exported. The JSON file includes the text as well as the associated annotation labels, tags, and history of changes of each label (including date, time, and author).	
Primary Actor	Expert Annotator	
Secondary Actors	None	
Trigger	Clicking the ‘Export’ button	
Main Scenario	**Step**	**Action**
	1	User clicks ‘Export’

	2	System creates an input popup, and user enters a name that will be associated with the revisions
	3	System opens file explorer
	4	User optionally overwrites name and clicks ‘Save’
	5	JSON is saved in the Downloads folder
Extensions	**Step**	**Branching Action**
	2a	Name is invalid (contains invalid characters or > 50 characters): - System creates an error message, then prompts the user to enter another name. Does not let user export annotations unless a valid name is entered
	5a	File name already present in Downloads folder: - the file name will be postfixed with a number, i.e. annotations-10-29-23.json annotations-10-29-23(1).json and annotations-10-29-23(2).json
Open Issues	None	
References	Fig. 1.4.1, Fig. 1.4.2, Fig. 1.4.3	
Related Use Cases	All use cases. This use case is executed after every single other use case. However, since the current functionality already allows you to export the information (although the information is exported as three separate files), the same outcome can be achieved with the current application. This is with the exception of history, which is not tracked in the current application.	

ID	FR-3	
Name	Undo Action	
Summary	User undoes the last performed action, whether that be reversing - adding a label, deleting a label, replacing a label or - accepting a label, rejecting a label, suggesting a different label, or reopening a label. Repeated undo commands will keep rolling back actions in reverse chronological order until there are no more actions to undo.	
Priority	3	
Preconditions	The user has imported a file and is in the edit screen.	
Postconditions	The last-performed action will be undone, if it exists.	
Primary Actor	Expert Annotator	
Secondary Actors	None	
Trigger	Click the ‘Undo’ button	
Main Scenario	**Step**	**Action**
	1	The user clicks the undo button
	2	The system undoes the last performed action (from the list included in Summary).
Extensions	**Step**	**Branching Action**

	1a	There is no action to undo - The undo button will be grayed out, and will be unclickable
Open Issues	None	
References	Fig. 1.4.1, Fig. 1.4.2, Fig. 1.4.3	
Related Use Cases	FR-4, FR-5, FR-6, FR-7	

ID	FR-4	
Name	Replace label in Annotation Mode	
Summary	After uploading a file in Annotation Mode, the user shall be able to easily replace labels with a different label. There will be a ‘Replace’ button on each annotation that allows the user to change the label for that piece of text.	
Priority	5	
Preconditions	The user has text in the editing screen that is annotated with at least one label.	
Postconditions	The label for an annotation has been switched from one label to another. The change is recorded in the history, noting the new and old label, the change date, time, and author.	
Primary Actor	Expert Annotator	
Secondary Actors	None	
Trigger	The user clicks the ‘Replace’ button next to a label, which is likely to be a small icon.	
Main Scenario	**Step**	**Action**
	1	In Annotation Mode, the user uploads JSON annotations or uploads a text file and then creates annotations. There are at least two labels.
	2	The user clicks the ‘Replace’ icon next to a label in the text.
	3	The user selects a new label from the list at the top.
	4	The label is updated in accordance with their selection (including color) and the action is recorded in the history log
Extensions	**Step**	**Branching Action**
	3a	The user uses keyboard shortcuts and presses a number 0-9 - The respective label will be applied. (1 is the leftmost label, while 9 is the second-rightmost label, and 0 is the rightmost label). If there are more than 10 labels, this functionality will only work for the first 10.
	3b	The new label is the same as the old label - No change will be made. No changes will be logged to history.
Open Issues	None	
References	Fig. 1.4.2	
Related Use Cases	FR-3	

ID	FR-5	
Name	Accept or reject label in Review Mode	
Summary	After uploading annotations in Review Mode, the user marks a label as accepted or rejected.	
Priority	4	
Preconditions	The user has uploaded a JSON with annotations.	
Postconditions	The specific label will be marked as accepted or rejected, and will be considered completed. If exported, the export JSON will include the change next to the author, date, and time.	
Primary Actor	Expert Annotator	
Secondary Actors	None	
Trigger	Clicking on a label in review mode	
Main Scenario	**Step**	**Action**
	1	The user uploads annotations in Review Mode, and is in the editing screen.
	2	The user clicks the accept or reject button for some label.
	3	The label is marked as accepted or rejected, and the system records the change in the history log, so that if the user clicks 'Export', then the annotation will be marked as accepted or rejected by the name of the current user, as well as the date and time.
Extensions	**Step**	**Branching Action**
	2a	User clicks to accept or reject a label that is already accepted or rejected: - Nothing happens. The button will be grayed out and unclickable. No changes are made to display or history.
Open Issues	None	
References	Fig. 1.4.3	
Related Use Cases	FR-3	

ID	FR-6	
Name	Suggest label in Review Mode	
Summary	The user marks an annotation as needing a different label.	
Priority	3	
Preconditions	The user has uploaded annotations in Review Mode, and is in the editing screen.	
Postconditions	The specified annotation will be marked with a different label as a suggestion. The label will be considered completed.	
Primary Actor	Expert Annotator	
Secondary Actors	None	
Trigger	Clicking the suggest button on a label in Review Mode	
Main Scenario	**Step**	**Action**

	1	User uploads annotations in Review Mode, and has begun reviewing annotations in the editing screen.
	2	User clicks ‘Suggest’ for some label.
	3	The user selects a new label from the list at the top.
	4	The suggested label is updated in accordance with their selection and the action is recorded in the history log.
Extensions	**Step**	**Branching Action**
	3a	The user uses keyboard shortcuts and presses a number 0-9 - The respective label will be applied. (1 is the leftmost label, while 9 is the second-rightmost label, and 0 is the rightmost label). If there are more than 10 labels, this functionality will only work for the first 10.
	3b	The suggested label is the same as the original label - No change will be made. No changes will be logged to history. The label will be considered completed.
Open Issues	None	
References	Fig. 1.4.3	
Related Use Cases	FR-3	

ID	FR-7	
Name	Reopen label in Review Mode	
Summary	After the user has accepted, rejected, or suggested another label, the user goes back and reopens the label. The label is then considered unfinished, and can once more be accepted, rejected, or suggested to switch to another label.	
Priority	4	
Preconditions	User is in review mode, and has accepted, rejected, or suggested a different label for some annotation.	
Postconditions	Annotation will need to be reviewed again by clicking ‘Accept’, ‘Reject’ or ‘Suggest’ icons. The label will not be considered completed.	
Primary Actor	Expert Annotator	
Secondary Actors	None	
Trigger	Clicking the ‘Suggest’ button on a label in Review Mode	
Main Scenario	**Step**	**Action**
	1	The user uploads annotations to review in Review Mode.
	2	The user starts reviewing labels. They realize that they made a mistake a couple of labels back, but they don’t want to undo their recent changes in order to fix the problem.
	3	The user clicks the ‘Reopen’ button on the label they want to change.
	4	The user accepts, rejects, or suggests another label for the annotation.
	5	The user resumes reviewing from their previous location.

Extensions	**Step**	**Branching Action**
	3a	User attempts to reopen an unchanged label: - 'Reopen' button is not clickable, and is grayed out
Open Issues	None	
References	Fig. 1.4.3	
Related Use Cases	FR-3	

2.2 Functional Requirement Testing

This section includes tests that thoroughly verify that the added functionality works as intended. Each test has a summary scenario, then the preconditions, steps, and expected results that would test the scenario. It also references which functional requirement it tests from above. This correlation is also visible in the test numbers.

Test 1.1: Import JSON annotations by selecting file

Tests: FR-1

Scenario: Verify that the user is able to successfully import a valid JSON file using the file explorer.

Precondition: The user is on the home screen.

Test Steps:

- Under 'Review annotations', click 'Import.'
- System opens the file explorer.
- Select a valid JSON file with annotations.
- Click 'Import.'

Expected Results: The system successfully imports the JSON file, and the text with annotations and correctly colored tags is displayed.

Test 1.2: Import JSON annotations by dragging

Tests: FR-1

Scenario: Verify that the user is able to successfully import a valid JSON file by dragging.

Precondition: The user is on the home screen.

Test Steps:

- Drag a valid JSON file with annotations into the box below 'Review annotations'

Expected Results: The system successfully imports the JSON file, and the text with annotations and correctly colored tags is displayed.

Test 1.3: Import non-JSON annotations file

Tests: FR-1

Scenario: Verify that if the user tries to upload a non-JSON annotations file, the system will raise an error message and not import it.

Precondition: The user is on the home screen.

Test Steps:

- Under 'Review annotations' click 'Import.'
- System opens the file explorer.

- Select a file to upload that is not of JSON format.
- Click 'Import.'

Expected Results: The system will display an error message saying: ‘invalid format, please upload JSON’

Test 1.4: Import invalid JSON annotations file

Tests: FR-1

Scenario: Verify that if the user tries to upload an invalid JSON file, the system will raise an error message and not import it.

Precondition: The user is on the home screen.

Test Steps:

- Under ‘Review annotations’ click 'Import.'
- System opens the file explorer.
- Select a file to upload that is JSON format, but contains invalid syntax (extra open and close square brackets, and or invalid characters)
- Click 'Import.'

Expected Results: The system will display an error message saying: ‘invalid JSON file, error on line <line>’

Test 2.1: Export JSON with valid name

Tests: FR-2

Scenario: Verify that after the user creates annotations they can successfully export it, provided they enter a valid name.

Preconditions: The user is on the home screen.

Test Steps:

- On the home screen, drag a valid JSON file with annotations into the box below ‘Review annotations’
- Create accept, reject and suggest different labels for multiple annotations
- Click ‘Export’ on the top bar
- Enter a valid name (<50 characters, UTF-8 characters) and press ‘Export’

Expected Results: JSON is saved to the downloads folder. This JSON includes all of the changes made by the user, and notes the history of who made the annotations, the date, and the time, on each change.

Test 2.2: Export JSON with invalid name

Tests: FR-2

Scenario: Verify that after the user creates annotations they cannot successfully export it if they have not entered a valid name.

preconditions:

Test Steps:

- On the home screen, drag a valid JSON file with annotations into the box below ‘Review annotations’
- Create accept, reject and suggest different labels for multiple annotations
- Click ‘Export’ on the top bar
- Enter an invalid name (test with both >50 characters and non-UTF-8 characters) then press ‘Export’

Expected Results: No JSON is saved. System produces an error message explaining that there were either >50 characters or non-UTF-8 characters in the name the user entered. Nothing else on the page is clickable while this popup is in place. Once the user clicks ‘OK’ on this popup, the ‘enter name’ popup shows again.

Test 3.1: Undo actions in Annotation Mode

Tests: FR-3

Scenario: Verify that the user is able to undo a previous action in Annotation Mode, including adding a label, deleting a label, and replacing a label. To avoid redundancy and bulk, every Annotation Mode action is included in this one test. With guidance from Dr. Gurney, we consider this acceptable, since implementation-wise, the undo action will have the same functionality no matter the action, given that actions will be stored as a queue, and the queue is popped when someone clicks ‘Undo’.

Precondition: User has uploaded JSON or text file in Annotation Mode.

Test Steps:

- Add a label, verifying that the label was added
- Click Undo, verifying that the label is removed
- Add a label, verifying that the label was added
- Delete a label, verifying that the label is deleted
- Click Undo, verifying that the label is re-added
- Click Replace on the label and select a different label, verifying that the label was replaced
- Click Undo, verifying that the annotation returns to its original label
- Click ‘Export’, and enter a valid name

Expected Results: After each change, the last made change will be reverted, and the text in GUI will be the exact same as it was before the last change was made. In the exported document none of these changes will be logged (i.e. undo actions will not be logged in the history–they will simply delete the previous action).

Test 3.2: Undo actions in Review Mode

Tests: FR-3

Scenario: Verify that the user is able to undo a previous action in Review Mode, including accepting a label, rejecting a label, suggesting a different label, and reopening a label. To avoid redundancy and bulk, every Review Mode action is included in this one test. With guidance from Dr. Gurney, we consider this acceptable, since implementation-wise, the undo action will have the same functionality no matter the action, given that actions will be stored as a queue, and the queue is popped when someone clicks ‘Undo’.

Precondition: User has uploaded JSON or text file in Review Mode.

Test Steps:

- Accept a label, verifying that the label was accepted and considered completed
- Click Undo, verifying that the label is label is not marked as accepted, and not considered completed
- Reject a label, verifying that the label was rejected and considered completed
- Click Undo, verifying that the label is label is not marked as rejected, and not considered completed

- Suggest a different label for a label, verifying that the suggestion was shown and the annotation was considered completed
- Click Undo, verifying that the suggestion is no longer shown and the annotation is not considered completed
- Click Reopen on a label that you had accepted earlier, verifying that the label no longer appears as accepted, and is not considered completed
- Click Undo, verifying that the annotation appears as accepted and is considered completed
- Click Reopen on a label that you had rejected earlier, verifying that the label no longer appears as rejected, and is not considered completed
- Click Undo, verifying that the annotation appears as rejected and is considered completed
- Click Reopen on a label for which you had suggested a different label earlier, verifying that the label no longer includes the suggestion, and is not considered completed
- Click Undo, verifying that the suggestion reappears and the label is considered completed
- Click 'Export', enter a valid name, and click 'Export'

Expected Results: After each change, the last made change will be reverted, and the text in GUI will be the exact same as it was before the last change was made. Since there are no more changes to undo, the undo button will be grayed out and unclickable. In the exported document none of these changes will be logged (i.e. undo actions will not be logged in the history–they will simply delete the previous action).

Test 4.1: Replace label in Annotation Mode with a new label

Tests: FR-4
Scenario: Verify that a user is able to replace a label with another in Annotation Mode.
Precondition: The user is in Annotation Mode, and has uploaded a JSON with text and defined labels.
Test Steps:

- On some tag, click the 'Replace' icon
- Click a new label that will replace the current label. Also test this using the keyboard shortcuts, pressing a number instead of clicking a label
- Click 'Export', enter a valid name, and click 'Export'

Expected Results: The annotated text within the GUI will display the correct color labeling with the new annotation labeling. In the exported JSON, the change is recorded for the given annotation, including the name, date, and time.

Test 4.2: Replace label in Annotation Mode with the same label

Tests: FR-4
Scenario: Verify that if a user replaces a label with the same label in Annotation Mode, a warning will pop up, giving the user a choice–either 'choose a different label' or 'keep label:<label>'.
Precondition: The user is in Annotation Mode, and has uploaded text or JSON.
Test Steps:

- On some annotation, click the 'Replace label' icon

- Click the same label as the current label. Also test this using the keyboard shortcuts, pressing a number instead of clicking a label
- In the popup, click ‘choose a different label’ and select a different label
- On some other annotation, click the ‘Replace label’ icon
- Click the same label as the current label. Also test this using the keyboard shortcuts, pressing a number instead of clicking a label
- In the popup, click ‘keep label: <label>’
- Click ‘Export’, enter a valid name, and click ‘Export’

Expected Results: The annotated text within the GUI will indicate that the user has replaced a label with a new label for the first annotation (updating the corresponding color). For the second label, the GUI will not have changed. The system considers both annotations as completed. In the JSON output, the new replaced label is present, with the name, date, and time of the change associated with the replacement. The unchanged label is also present, but does not include the replacement as a change, and thus does not list the associated name, date, and time.

Test 5.1: Accept/Reject labels in Review Mode

Tests: FR-5

Scenario: Verify that the user is able to successfully accept and reject labels in Review Mode.

Precondition: User has uploaded a JSON with text and defined labels to the system in Review Mode.

Test Steps:

- Click the ‘Accept’ icon on a label.
- Verify that the label indicates it is accepted (possibly grayed out, or with the accept icon colored differently) and that the system considers it as completed
- Click the ‘Reject’ icon on a label
- Verify that the label indicates it is rejected (possibly grayed out, or with the reject icon colored differently) and that the system considers it as completed
- Click ‘Export’, enter a valid name, and click ‘Export’

Expected Result: In the exported JSON, the corresponding labels are indicated as accepted and rejected. The history of these actions is included, with the name, date and time of the change.

Test 6.1: Suggest different label in Review Mode

Tests: FR-6

Scenario: Verify that a user is able to suggest a different label in Review Mode.

Precondition: The user is in Review Mode, and has uploaded a JSON with text and defined labels.

Test Steps:

- On some annotation, click the ‘Suggest different label’ icon
- Click a new label that will replace the current label. Also test this using the keyboard shortcuts, pressing a number instead of clicking a label
- Click ‘Export’, enter a valid name, and click ‘Export’

Expected Results: The annotated text within the GUI will indicate that the user has suggested a new label. This may mean that the label is grayed out, or has the ‘Suggest’ icon colored differently. The system considers the specific annotation as completed. In the JSON output, the suggested label is present, with the name, date, and time of the change associated with the suggestion.

Test 6.2: Suggest the same label in Review Mode

Tests: FR-6

Scenario: Verify that if a user suggests the same label in Review Mode, a warning will pop up, giving the user a choice–either 'choose a different label' or 'accept label:<label>'.

Precondition: The user is in Review Mode, and has uploaded a JSON with text and defined labels.

Test Steps:

- On some annotation, click the 'Suggest different label' icon
- Click the same label as the current label. Also test this using the keyboard shortcuts, pressing a number instead of clicking a label
- In the popup, click 'choose a different label' and select a different label
- On some other annotation, click the 'Suggest different label' icon
- Click the same label as the current label. Also test this using the keyboard shortcuts, pressing a number instead of clicking a label
- In the popup, click 'accept label: <label>' and select a different label
- Click 'Export', enter a valid name, and click 'Export'

Expected Results: The annotated text within the GUI will indicate that the user has suggested a new label for the first label. For the second label, the GUI will indicate that the user has accepted the label. Either of these may mean that the label is grayed out, or has the 'Suggest' or 'Accept' icon colored differently. The system considers the specific annotations as completed. In the JSON output, the new suggested label is present, with the name, date, and time of the change associated with the suggestion. The accepted label is also present, with the associated name, date, and time.

Test 7-1: Reopen annotation in Review Mode

Tests: FR-7

Scenario: Verify that the user can reopen a label marked as accepted, rejected, or suggested in review mode. Similarly to Test 3.1 and 3.2 testing the 'Undo' button, since 'Reopen' does the same thing no matter what the previous annotation was, we have kept this as a single test.

Precondition: The user has uploaded annotations in Review Mode.

Test Steps:

- Accept a label
- Reject a label for a different annotation
- Suggest a different label for another annotation
- Click the 'Reopen' icon for each label
- Accept all three labels, and accept any other labels in the document
- Click 'Export', enter a valid name, and click 'Export'

Expected Result: After pressing 'Reopen', the GUI will indicate that each annotation must be accepted, rejected, or suggested once again. After accepting all annotations and exporting, the Reopen actions will not be included in the JSON output. Only the final choices will be included with their respective authors, dates, and times.

3. Non-Functional Requirements

3.1 Summary of Non-Functional Requirements

This section includes the non-functional requirements, which describe the performance of the product, rather than the functionality. We include topics such as supported browser versions, fault tolerance, export time, data protection, load handling, accessibility, and ease of use. Each test is given a priority. A rating of 5 means it is the most essential, while a rating of 1 means it is the least essential.

3.1 List of Non-Functional Requirements

NFR Number: NFR-0	**Priority:** 5
Description: The web app must support running on Chromium 80.0.3987.33 or later.	
Tests: Access the application and perform annotation and review, imports and exports, on Chromium 80.0.3987.33, as well as the latest version of Chromium (Chromium Releases - GitClear, n.d.).	

NFR Number: NFR-1	**Priority:** 5
Description: The system must perform without failure in 95 percent of use cases during a 30-day period.	
Tests: Perform every use case described in this document once per day for 30 days. Each day, record the percent of use cases that passed, then average at the end of the 30 days.	

NFR Number: NFR-2	**Priority:** 4
Description: The system shall not take longer than 1 second to export a JSON file of 1 KB or less.	
Tests: Export a JSON file of 1KB, noting the time it takes to export using browser development tools.	

NFR Number: NFR-3	**Priority:** 1

Description: The application must comply with the General Data Protection Regulation (GDPR) in terms of data handling and user consent.
Tests: Clearly document how the application handles data. Ensure this documentation is genuine to how the application actually works. Obtain user consent if necessary. Ensure these methods are in line with the GDPR.

NFR Number: NFR-4	**Priority:** 2
Description: The application should be able to handle a concurrent amount of 50 or fewer users without fault.	
Tests: Using Selenium, create 50 bots that access the site at the same time and perform miscellaneous operations including annotating and reviewing, importing and exporting. Monitor the response times and database queries.	

NFR Number: NFR-5	**Priority:** 2
Description: The web application shall adhere to WCAG 2.0 Level AA accessibility standards, ensuring it is usable by people with disabilities.	
Tests: Clearly document how the application is usable for people with disabilities. This documentation should be genuine to how the application actually works. Ensure its operation is in line with WCAG 2.0 Level AA standards.	

NFR Number: NFR-6	**Priority:** 1
Description: All code shall follow the style used within the original application	
Tests: Ensure that code formatting matches that of the original application. The tab sizing should be the same, the bracket spacing and line formatting should match, and variable naming styles should match. This will allow easy continued work on the open-source project.	

NFR Number: NFR-7	**Priority:** 4
Description: The application shall include a link to instructions for usage that are able to teach someone how to use the application within 10 minutes.	
Tests: Recruit someone who has not used this application before and have them read the instructions. Ask them to indicate when they feel that they have a complete understanding of how to use the application. This test will pass if that occurs within 10 minutes.	

NFR Number: NFR-8	**Priority:** 4

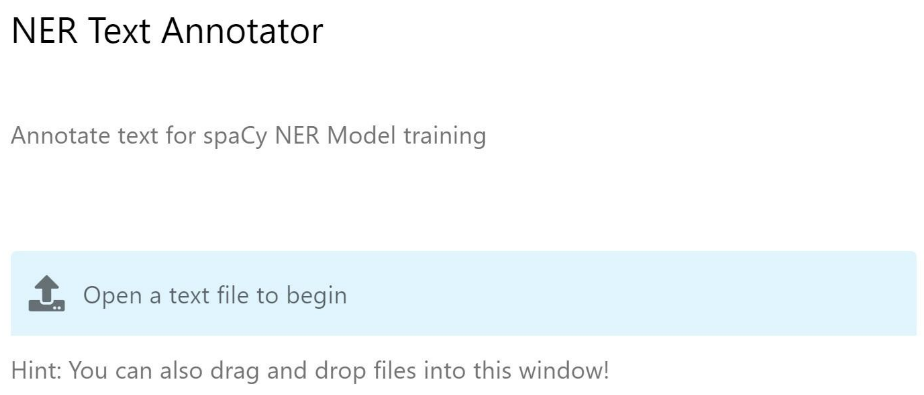
Description: The application should load and be made available within 4 seconds of opening.
Tests: Using browser development tools, open the site and ensure that the page loads within four seconds.

NFR Number: NFR-9	**Priority:** 4
Description: The application's additional user interface elements should match existing elements stylistically.	
Tests: Ensure that all newly added colors, shapes, fonts, and other traits match the original application. This is difficult to test, but could be achieved using a third party. For example, headings should stay the same color, rounded radius should be the same for annotations, and the simplistic, modern style should be preserved.	

NFR Number: NFR-10	**Priority:** 5
Description: The application shall be compatible with a variety of sized screens	
Tests: Load the site on mobile and desktop. Load the site using the smallest window size possible and the largest window size possible. Ensure all elements appear legibly and nothing overlaps such that the application is still usable and intuitive.	

4. User Interface

The "User Interface Design Document for NER Text Annotation Manager" is yet to be created. However, this section includes a few mockups of the anticipated GUI. Figure 4.1 is the old home screen, and Figure 4.2 shows a proposed new home screen. Figure 4.3 shows the old editing screen, while Figure 4.4 shows a proposed new editing screen in Review Mode. Annotation mode would be very similar to Review Mode visually, so it was not included here. The tool icons and locations have not been visualized yet.

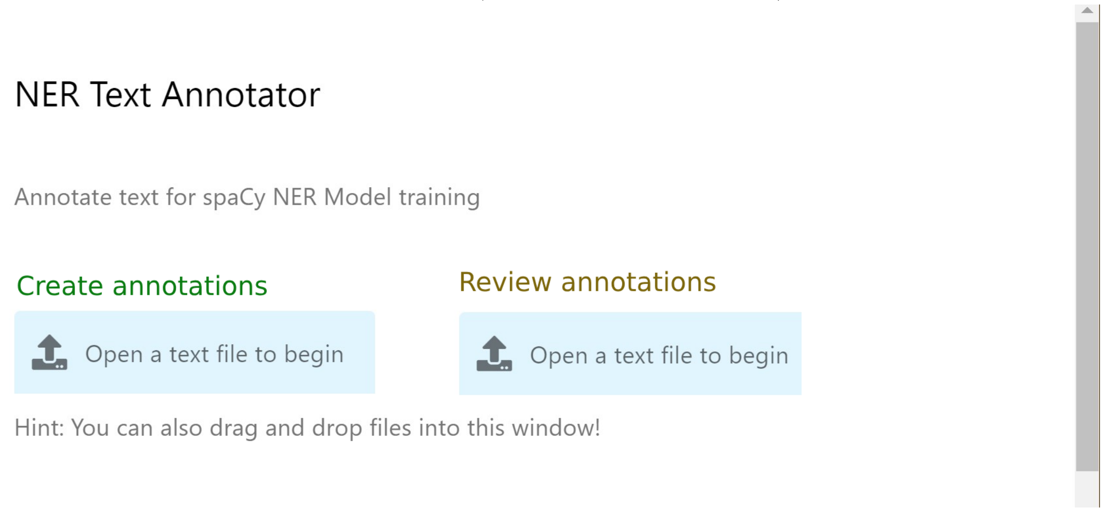

NER Text Annotator

Annotate text for spaCy NER Model training

Open a text file to begin

Hint: You can also drag and drop files into this window!

Figure 4.1: Old home screen without new modes (Arunmozhi, 2020/2023)


NER Text Annotator

Annotate text for spaCy NER Model training

Create annotations

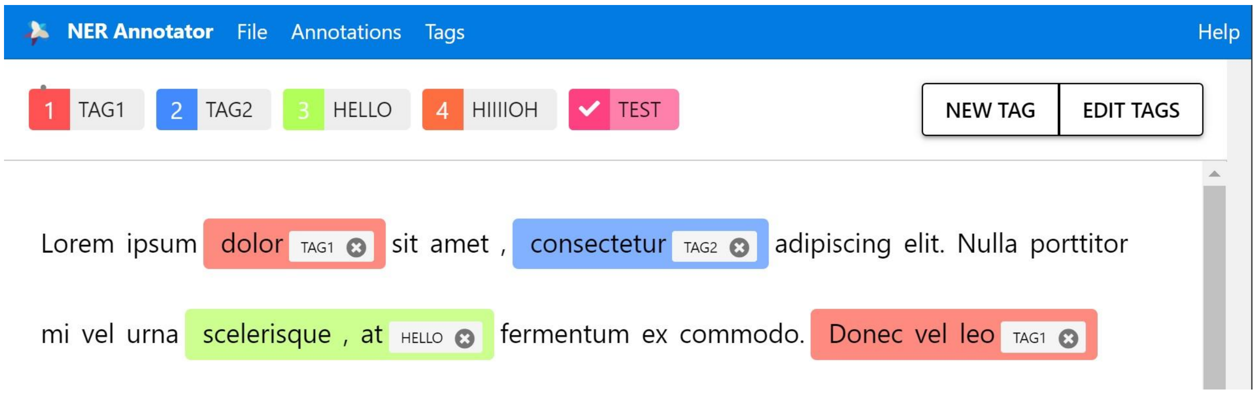
Open a text file to begin

Review annotations

Open a text file to begin

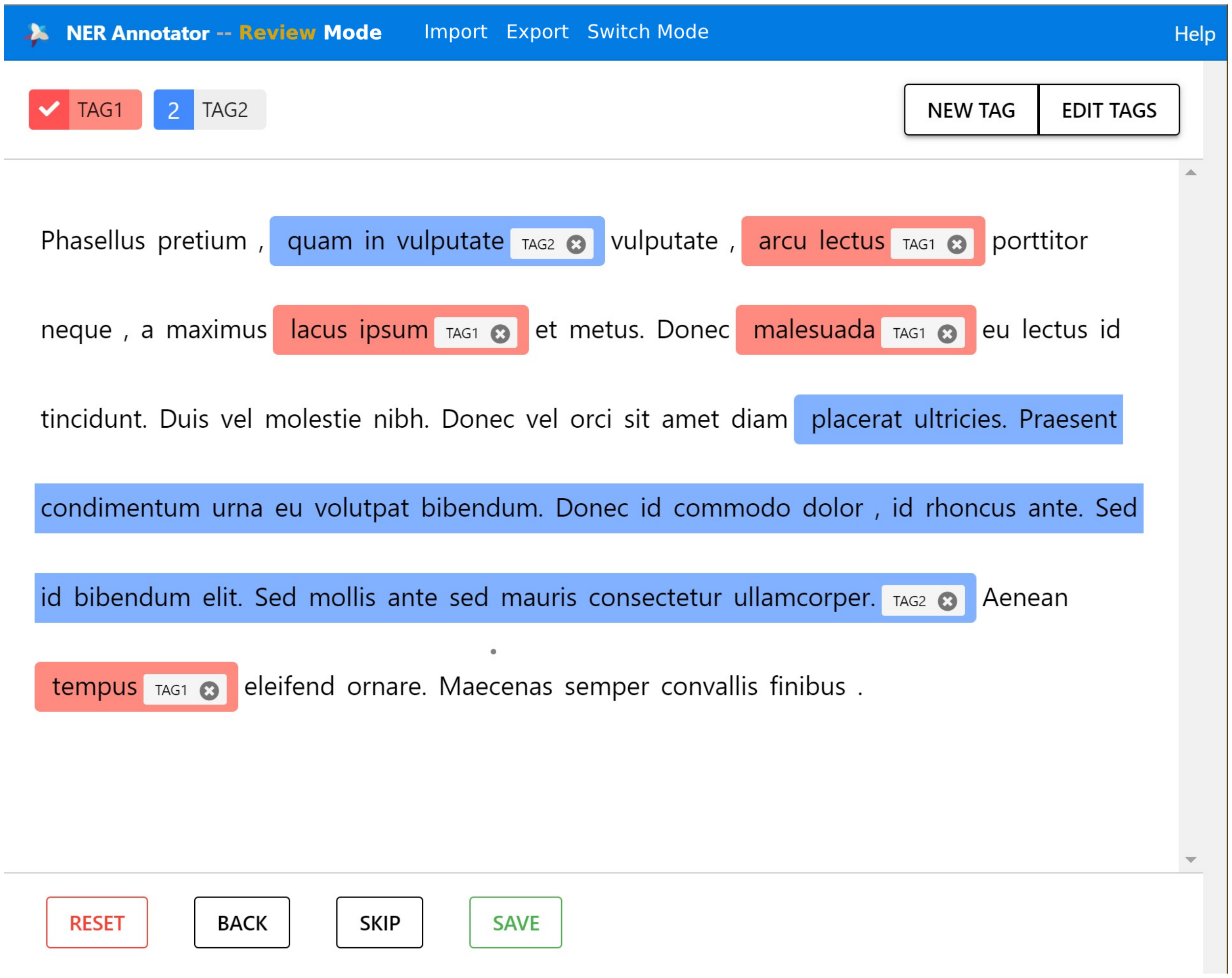
Hint: You can also drag and drop files into this window!

Figure 4.2: New home screen, with both modes



Lorem ipsum dolor TAG1 sit amet , consectetur TAG2 adipiscing elit. Nulla porttitor mi vel urna scelerisque , at HELLO fermentum ex commodo. Donec vel leo TAG1

Figure 4.3: Old edit screen (Arunmozhi, 2020/2023)



Phasellus pretium , quam in vulputate TAG2 vulputate , arcu lectus TAG1 porttitor neque , a maximus lacus ipsum TAG1 et metus. Donec malesuada TAG1 eu lectus id tincidunt. Duis vel molestie nibh. Donec vel orci sit amet diam placerat ultricies. Praesent condimentum urna eu volutpat bibendum. Donec id commodo dolor , id rhoncus ante. Sed id bibendum elit. Sed mollis ante sed mauris consectetur ullamcorper. TAG2 Aenean tempus TAG1 eleifend ornare. Maecenas semper convallis finibus .

Figure 4.4: New Review Mode screen (changes made to top navigation bar)
Note: Annotation Mode will be similar, except in green text color instead of brown/yellow

5. Deliverables

This section describes the deliverables that will be available by the completion of this project.

The following deliverables will be available by May 2nd, 2023.

Hard copies of each of the following:

- Systems Requirement Specification: [November 1st, 2023]
- System Design Document: [November 15th, 2023]
- User Interface Design Document: [November 29th, 2023]
- User Manual
- Administrator Manual
- Copies of all Biweekly Status Reports

An electronic file containing the following:

- Systems Requirement Specification: [November 1st, 2023]
- System Design Document: [November 15th, 2023]
- User Interface Design Document: [November 29th, 2023]
- User Manual
- Administrator Manual
- All source code
- The executable program
- Any other software required for installation and execution of the delivered program.

URL for the following webpages:
- Github Repository of source code

6. Open Issues

Client would like to wait until we meet again to sign the Agreement between customer and Contractor

Appendix A – Agreement Between Customer and Contractor

The agreement between the customer and client confirms that the contents of this document (SRS) is correct and agreed upon by both parties. This is a mutual agreement between the customer (NER Text Annotation Manager) and the project team (SEWDO) regarding the outlined project requirements. Both parties acknowledge the specifications detailed within the document, encompassing the modification of the web-based NER annotation tool, the creation of the "Review Mode", and the ability to import and export annotation in a single file in JSON format. All parties agree that these requirements have been discussed and understood and to be executed as described.

In the event that changes are to be made in the future to this document, a formal change request procedure will be followed. The customer or any member of the project team may propose modifications by submitting a change request in writing. The request will be reviewed and evaluated by the project team, and if approved, the document will be updated accordingly. all changes will be documented and communicated to all relevant parties. This goes into effect upon the signing of this document by all parties.

The agreement can be reached below and at the following URL:
https://docs.google.com/document/d/1NfrvzclqWs2lCYTkhmWJ7K6c9zCVMDYivBSXNJc7WAM/edit?usp=sharing

Note: Our client would like to discuss this SRS with us before signing it. We sent this document to him on Monday 10/30, and have scheduled to talk with him next week about it.

Signature (Client)

x ______________________	Name (Typed)________________	Date __________
x ______________________	Name (Typed)________________	Date __________

Signature (Project Team)

x *Wilder Baldwin*	Name (Typed) Wilder Baldwin	Date 10/30/23
x *Samuel Waggoner*	Name (Typed) Samuel Waggoner	Date 10/30/23
x *Darien Orethun*	Name (Typed) Darien Orethun	Date 10/30/23
x *Owen Bellew*	Name (Typed) Owen Bellew	Date 10/30/23

Comments:

Appendix B – Team Review Sign-off

All members of team SEWDO, have diligently reviewed the contents, requirements and specifications of the SRS document and are in unanimous agreement with its specifications. There are no points of contention within the team, but in the event that they may arise, the space provided below will be used as a common ground for minor points or suggestions. This will ensure that even the smallest constructive criticisms don't go unnoticed and such comments are resolved in a timely manner. This will enhance clarity and accuracy of our development process and should be a useful and frequent tool to alleviate any contentions.

The agreement can be reached below and at the following URL:
https://docs.google.com/document/d/1BqsTQXbSDKOGeY7RtVXBdN5QmcYROT2x4JhUKC8-qrY/edit?usp=sharing

Signature (Project Team)

x *Wilder Baldwin*	Name (Typed) Wilder Baldwin	Date 10/30/23
x *Owen Bellew*	Name (Typed) Owen Bellew	Date 10/30/23
x *Sam Waggoner*	Name (Typed) Samuel Waggoner	Date 10/30/23
x *Darien Orethun*	Name (Typed) Darien Orethun	Date 10/30/23

Appendix C – Document Contributions

Sam Waggoner 35%

- Contributed to all sections in introduction
- Created use case diagrams
- Wrote the NFR tests
- Reworked all FRs
- Reworked all FR tests
- Created mockups for the UI section

Darien Orethun 25%

- Contributed to NFRs
- Revised the deliverables section
- Contributed to FRs
- Contributed to the 1.x sections

Wilder Baldwin	20%

- Contributed to FRs
- Contributed FR tests
- Revised section 1

Owen Bellew	20%

- Appendix A
- Appendix B
- Contributed to NFRs