ACCESSIBILITY ON THE WEB THROUGH SEMANTIC AND SOCIAL RENARRATIONS

by

Emre Hoş

B.S., Mechatronics Engineering, Yıldız Technical University, 2016

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Systems and Control Engineering

Boğaziçi University

2021

ACKNOWLEDGEMENTS

I would like to thank my supervisors, Prof. Yağmur Denizhan and Dr. Suzan Üsküdarlı, for their invaluable support, guidance, thoughtful comments, and recommendations on this work. I would specially thank Dr. Suzan Üsküdarlı whose insight and knowledge into the subject matter guided me through this work.

I would like to thank the members of my thesis committee, especially Assoc. Prof. Venkatesh Choppella for his kindness and participation.

I would also like to sincerely thank to friends, relatives and music.

Finally, I would like to express my deep and sincere gratitude to my family for their unconditional love and support. This work would not have been possible without them, and I dedicate this milestone to them.

ABSTRACT

ACCESSIBILITY ON THE WEB THROUGH SEMANTIC AND SOCIAL RENARRATIONS

The Web was proposed in 1989 with the vision of being an open platform that everyone can access. This vision of providing accessibility to all people is one of the primary goals of the Web. To this end, the Web Accessibility Initiative (WAI) was launched by W3C in 1997 mainly to improve the accessibility of Web for the people facing physical accessibility barriers. While such work is crucial, it falls short in delivering true accessibility, since further barriers to understanding the information exist. Such barriers are associated with the ability of a person to understand the information that reaches them, such as language, literacy, culture, and expertise. To overcome such barriers, this work utilizes the renarration method, the process of forming an alternative version for a pre-existing web document to reach alternative audience.

Accordingly, this work proposes an ontology that represents renarrations, web documents, and social interactions of renarrators and renarrations, and interconnections between them to support creation of an ontology driven social renarration platform. A prototype that utilizes Solid, a web decentralization project, has been created to demonstrate the feasibility of the social renarration approach and the suitability of the ontology for supporting the realization of a decentralized social renarration platform that protects the provenance and privacy.

ÖZET

ANLAMSAL VE SOSYAL YENİDEN ANLATIMLAR İLE WEB'DEKİ ERİŞİLEBİLİRLİK

Web, herkesin erişimine açık bir platform olma vizyonu ile 1989 yılında tasarlandı. Bu vizyon, Web'in temel amaçlarından biri olan herkesin erişiminin sağlanması ile ilgilidir. Bu maksatla, esas olarak fiziksel erişim engellerine sahip insanların Web'e erişimini iyileştirebilmek adına Web Accessibility Initiative (WAI) 1997 yılında faliyete geçirildi. Bu çalışmalar her ne kadar büyük önem arz etse de erişilebilirlik konusunu bütünüyle ele almak konusunda yeterli gelmemektedir çünkü bilgiyi anlamak konusunda daha öte engeller de bulunmaktadır. Bunlar, bilgiyi anlama konusundaki yeterlilik ile ilişkili olan, kişinin dili, eğitimi, kültürü ve uzmanlığı gibi meselelere dair engellerdir. Bu çalışma bu tür engelleri aşmak için yeniden anlatım (renarration) yöntemini, önceden var olan bir web dokümanının alternatif kişilere erişmek amacıyla alternatif varyantını biçimlendirme işlemi, kullanmaktadır.

Dolayısıyla, bu çalışma, yeniden anlatımları, web dökümanlarını, ve yeniden anlatımları oluşturan kişiler ve yeniden anlatımlar arasındaki sosyal etkileşimleri, ve bütün bunlar arasındaki bağlantıları, ontoloji temelli sosyal yeniden anlatım platformu oluşturabilmeyi desteklemek adına, temsil edebilen bir ontoloji öne sürmektedir. Sosyal yeniden anlatım yaklaşımının uygulanabilirliğini ve ontolojinin, kaynağın ve gizliliğin korunduğu merkezi olmayan sosyal yeniden anlatım platformunu gerçekleştirebilmeyi destekleme konusundaki, kullanılabilirliğini gösteren bir prototip, Web'in merkezsizleştirilmesi projesi olan Solid'den faydalanılarak, geliştirilmiştir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS/ABBREVIATIONS

CSS	Cascading Style Sheets
DC	Dublin Core Elements
DOM	Document Object Model
FOAF	Friend Of A Friend Vocabulary Specification
HTML	Hypertext Markup Language
OWL	Web Ontology Language
RDF	Resource Description Framework
RDFS	Resource Description Framework Schema
SPARQL	SPARQL Protocol and RDF Query Language
Turtle	Terse RDF Triple Language
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
W3C	World Wide Web Consortium
XSD	XML Schema Definition

1. INTRODUCTION

The vision of the Web is to make information available for everyone around the world [1]. Availability of the information requires that people have physical access to the devices and infrastructure that delivers web contents as well as the physical means for perceiving the content. Towards this end, much attention has focused on the infrastructure necessary to make Internet reachable to wider audiences. The technological advances in devices and telecommunication infrastructure has made significant advances in this regard. According to statistics, the percentage of the world population using the internet has reached 62% in 2020 (4.8 billion people) from 0.4% (16 million people) in 1995 [2]. To facilitate true accessibility, however, further barriers that inhibit the comprehension of web content must be addressed.

To address web accessibility issues, World Wide Web Consortium (W3C) launched the Web Accessibility Initiative (WAI) [3] [1]. WAI mainly focuses on standards to produce accessible content for the people facing physical accessibility barriers, such as vision disabilities. While the work of WAI is very important, it does not address accessibility issue stemming from comprehensibility, namely *semantic accessibility*. Such accessibility issues arise from language, culture, literacy, and expertise which must be addressed to render Web content truly accessible around the world.

To illustrate semantic accessibility, consider the difficulty experienced by a novice person in comprehending texts that includes complex mathematical concepts without any supporting visual material. One of the advantages of user generated content that has been supported since Web 2.0 [4] is the emergence of people who create content that provides alternative narrations of existing content. Such platforms include blogs (Nicaragua's Rural Youth on a blog [5]), domain-specific platforms (First Information Report that provides easy to understand illustrations for reporting crime in India by a contributor on a website [6]), and video platforms (Fourier Series by Khan Academy on Youtube [7]). The act of providing an alternative narration for some material can be referred to as renarrating. Achieving this by such platforms is fairly easy, however

they do not explicitly capture the original content nor the transformations. Such issues are important concerns for tracking the provenance and the specific transformations for the purpose of credibility.

To provide support for renarrations several approaches have been proposed. These approaches provide mechanisms for representing when and by whom a renarration was created for a given source. Some approaches have utilized meta-data for this purpose [8] [9]. A more formal approach was introduced in Renarration Data Model [10] [11], which introduces an ontology-driven approach for defining renarrations. For this purpose they specify an ontology that represents the source, creator, and renarration transformations. The utility of this ontology in specifying renarrations was demonstrated with a stand-alone application. While this ontology specifies the core concepts of renarration it does not address the concepts of a renarrator community, which we believe to be significant in increasing accessibility.

In this work we propose an ontology driven social renarration platform for creating and discovering renarrations and enabling social interactions. This platform aims to preserve the provenance of renarrated content, protect the privacy of renarrators, support interaction, and provide recommendations to increase accessible content. The explicit representation of the provenance protects web content developers from infringements of their rights as well as allows users to be informed about and appreciate the renarrators. Social interactions provide feedback on renarrations and provide the basis for determining the quality and trust in content and its providers. Supporting privacy and determining the access rights to one's own content allows the renarrators to decide how the content they have renarrated may be accessed. The ability to restrict access to sensitive material can be important under various personal or cultural contexts.

To support the social renarration platform, we introduce an ontology named Renarration Social Ontology [12] that represents renarrations, renarrators, and renarration related social interactions. As is consistent with the best practices of developing ontologies, various concepts and relations which are defined in W3C recommended ontologies were used in this ontology. The main influence in our work comes from the

Renarration Data Model, which has specified the core concepts of renarration. Additionally the W3C recommended standards Web Annotation Data Model [13] and Activity Streams 2.0 [14] were utilized.

A prototype of the proposed method is developed as a browser extension [15] (RNEx) as proof of concept. RNEx [16] facilitates the creation of renarrations, preserves the provenance of renarrated content, supports privacy and gives control of access rights to the owners of the renarrations. It also supports interaction, where users can see the body of work of renarrators, follow renarrators, rate and comment on renarrations. The prototype utilizes the web decentralization project Solid (Social Linked Data) [17] for storing and processing the semantically structured data related to renarrations and social interactions. The decentralization aspect is handled with the Solid project that is based on pods (personal online data stores). The data in pods are represented in RDF [18] (Resource Description Framework) to support semantic processing. The privacy protection aspect is also addressed via Solid which supports full control of the access rights to the owners [19] [20].

The main contributions of the work are:

(i) a model for a privacy protecting, decentralized, and ontology-driven social renarration platform to extend the semantic accessibility of web content
(ii) an ontology that represents renarrations, renarrators, and social interactions related to renarrations
(iii) a prototype (RNEx) to serve as a proof of concept for the proposed model that is implemented as a browser extension that utilizes the Solid to address the privacy and decentralization issues.
(iv) an evaluation that examines (1) the suitability (expressiveness) of the proposed ontology with test cases and (2) the feasibility of the proposed model through a use case with RNEx.

1.1. Structure

The remainder of the thesis is organized as follows. Chapter 2 presents information about the technologies and ontologies utilized throughout this work. Chapter 3 presents related work. Chapter 4 presents the requirements. Chapter 5 describes the proposed Renarration Social Ontology. Chapter 6 describes the prototype implementation which is a browser extension that utilizes ontologies and the Solid project to support decentralization and privacy. Chapter 7 expresses the evaluation process of the proposed Renarration Social Ontology and the prototype RNEx with various test cases and a use case. Chapter 8 indicates conclusions and provides ideas for future work.

2. BACKGROUND

This chapter presents the core knowledge that helped in the creation of the model and the prototype. It contains essential information on technologies, ontologies regarding the model and the prototype.

2.1. Technologies

2.1.1. Ontology

Ontology [21] [22] is the method of representing knowledge in a subject area. An ontology defines objects and relations in a particular domain of knowledge with terms, classes, categories, properties, and relations.

2.1.2. RDF

RDF [18] [23] [24] is a building block for ontologies. It provides a data-modeling vocabulary to describe web resources notionally, and a standard for representing the connection of web resources with triples that consist of URIs. RDF statements describe the web resources and the role of them in a triple. It expresses the elements of the triple as a subject, predicate, object with its *rdf:subject*, *rdf:predicate*, *rdf:object* properties. Furthermore, *rdf:type* property can be used to define the type of web resources, e.g. "*ex:Apple rdf:type ex:Fruit*". This triple declares that Apple is an instance of Fruit class.

2.1.3. RDF Schema

RDF Schema [24] [25] is a metadata data model that is based on the RDF model. It is used for describing the relation of the web resources notionally. It provides some fundamental elements to structure the ontologies, e.g. *rdfs:Class*, *rdfs:subClassOf*, *rdfs:domain*, *rdfs:range*, *rdfs:label*, *rdfs:comment*.

```
@prefix ex: <http://www.example.com/#>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.

ex:GrannySmith rdf:type ex:Apple.
ex:Cavendish rdf:type ex:Banana.
ex:Apple rdfs:subClassOf ex:Fruit.
ex:Banana rdfs:subClassOf ex:Fruit.

```

Figure 2.1. RDF Schema Example

The Figure 2.1 describes "GrannySmith is an Apple and Apple is a Fruit." , "Cavendish is a Banana and Banana is a Fruit." sentences with triples that utilizes RDF and RDFS models.

2.1.4. OWL

OWL [26] [27] is a semantic web language that enables defining logical relations between web resources. Ontologies state the axioms, constraints, and logical restrictions with the help of OWL. The OWL statements give additional and logical information about web resources because computer programs can interpret and utilize the logical statements to reveal and validate the obtained knowledge.

```
@prefix ex: <http://www.example.com/#>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
@prefix owl: <http://www.w3.org/2002/07/owl#>.

ex:isParentOf rdf:type owl:ObjectProperty.
ex:isChildOf rdf:type owl:ObjectProperty;
             owl:inverseOf ex:isParentOf.

ex:John rdf:type owl:NamedIndividual.
ex:Bob rdf:type owl:NamedIndividual.

ex:John ex:isParentOf ex:Bob.
```

Figure 2.2. OWL Example

In the Figure 2.2, it is noticeable that *ex:isParentOf* and *ex:isChildOf* are inverse properties. The "John is the parent of the Bob" sentence is declared with triples and "Bob is the child of John" sentence is not declared with triples. However, with the *owl: inverseOf* property in the "*ex:isChildOf owl:inverseOf ex:isParentOf.*" statement it can be inferred that "Bob is the child of John" since Owl defines logical relations

between web resources.

2.1.5. SPARQL

SPARQL [28] [29] [30] is a query language that enables obtaining and manipulating the structured data in the RDF format. SPARQL query contains triple patterns composed of subject predicate object elements that can be a variable. The outcome of the variables is retrieved by matching the query patterns in the dataset. SPARQL facilitates querying across the various data sources, endpoints e.g. graph databases, knowledge graphs, triplestores, etc.

```
1  @prefix foaf: <http://xmlns.com/foaf/0.1/> .
2
3  _:a foaf:name "Joe Darkoe".
4  _:a foaf:nick "TallJoe".
5  _:b foaf:name "Dave Hatela".
6  _:b foaf:nick "StrongDave".
7  _:c foaf:name "Jane Kindae".
8
```

Figure 2.3. Structured Data In Turtle Format

```
1 PREFIX foaf: <http://xmlns.com/foaf/0.1/>
2 SELECT ?name ?nick
3 WHERE {
4  ?x foaf:name ?name.
5  ?x foaf:nick ?nick.
6 }
```

Figure 2.4. SPARQL Query Example

Table 2.1. Query Result Table

name	**nick**
"Joe Darkoe"	"TallJoe"
"Dave Hatela"	"StrongDave"

Structured data in the Figure 2.3 is queried with the SPARQL query in the Figure 2.4. This query is designed to obtain the result of subject with *foaf:name* and *foaf:nick* properties. The results of the query can be seen in the Table 2.1.

2.1.6. Turtle

Turtle [31] [32] is a textual syntax for stating RDF data, and it is an alternative to N-Triples, JSON-LD, and RDF/XML, etc. formats. A triple in the Turtle syntax consists of a subject, predicate, and object, and these items express the role of web resources. Turtle triples represent knowledge in a machine-readable way.

```
1 @prefix ex: <http://www.example.com/#> .
2
3 ex:Cat a ex:Animal.
```

Figure 2.5. Turtle Example

The Turtle triple in the Figure 2.5 states that the "Cat" is the subject, "a" is the predicate and "Animal" is the object. This triple indicates that "Cat is an animal".

2.1.7. Chrome Extension

Chrome extensions [33] [34] are software programs that enable people to adjust behavior and functionality of their browsers. They utilize HTML [35], CSS [36], JavaScript [37] technologies to customize the experience.

2.1.8. Rdflib

Rdflib.js is a javascript library that is created to work with linked data in the Solid project [19]. Rdflib.js enables storing, fetching, tracking, parsing, serializing the structured data in the Solid pods. It is possible to read and write the data in the format of RDF/XML, Turtle, and read the data in the format of N3 and RDFa and JSON-LD, by using it. Lastly, it doesn't support creating full SPARQL queries but some SPARQL queries, e.g., graph match and optional [38] [39].

2.1.9. Solid-auth-client

Solid-auth-client [40] is a javascript library that allows apps to authenticate to Solid pods of users. It enables sending and retrieving data from the Solid [19] pods.

2.1.10. Solid

Solid [17] [19] (Social Linked Data) is a project that aims to re-decentralize the web. The decentralized design and architecture of Solid provide improved privacy and true data ownership for the individuals that use Solid based web applications. In Solid, user data is stored in pods (Personal Online Datastore), and data in the pod is independent of the Solid web applications that use it, owing to semantic web technologies. Users use their WebIDs, a decentralized identification mechanism, to log into the Solid account and Solid web applications. Solid utilizes LDP (Linked Data Platform) to organize the pods into containers and resources. Containers and resources in the Solid pods have their own URIs. Pods can store structured RDF data in Turtle, JSON-LD, or RDFa notation and some unstructured data, e.g., image, text. Users can fully control the containers and resources in their pods and decide to give read-write access to others through access control lists (ACLs). Lastly, Solid pod servers support live updates using WebSocket protocol, access control lists by adopting WebAccessControl(WAC) ontology, and optionally SPARQL.

2.1.11. WebID

WebID [41] [42] is a decentralized identification mechanism for uniquely identifying agents with URIs. It informs the agents about other agents that they are in communication with. It enables social networks between agents e.g., people, companies, organizations by allowing them to denote their relations with other agents. Protocols using WebID as an underlying method e.g., WebID-OIDC, WebID-TLS provide a new approach to authenticate internet services without passwords.

2.2. Utilized Ontologies

2.2.1. Web Annotation Data Model

Web Annotation Data Model [13] specification defines a model to share and reuse the annotation data across different software and hardware platforms. Annotation is

the operation of adding a piece of information that is related to the target fragment in the document. It can be thought of as connected resources that include a target and body. The Figure 2.6 shows the relation of body and target.

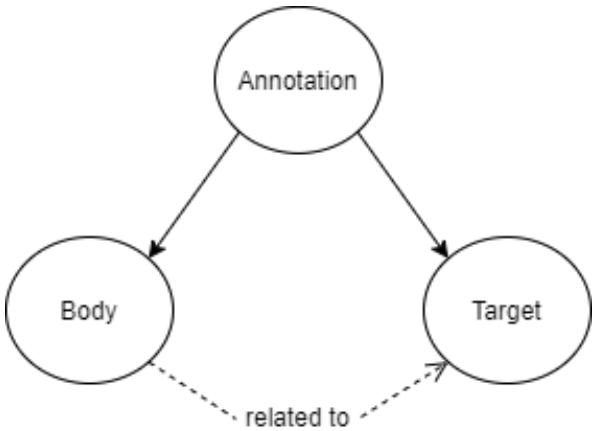

Figure 2.6. Depiction of a basic annotation

2.2.2. Activity Streams 2.0

Activity Streams 2.0 [14] specification defines a model for expressing the meta-data about activities in a rich, extensible, human-friendly, machine-readable way. It represents many types of actions "Create", "Read", "View", "Join", "Follow".

2.2.3. Friend Of A Friend Vocabulary

FOAF (Friend Of A Friend) [43] [44] is a semantic web vocabulary for describing people and their social networks. Information and people become connected with this machine-readable ontology. It is an example of a social semantic web application because of linking people, social groups, web documents, etc.

2.2.4. Dublin Core

Dublin Core [45] [46] is a metadata element set created for describing resources, and these resources can either be digital or physical resources, e.g., web pages, books. It has various elements for describing these resources, e.g., *dc:format*, *dc:language*, *dc:*

creator, *dc:title* etc.

2.2.5. XML Schema

XSD (XML Schema Definition) [47] [48] defines a specification for describing the XML document. It provides constraints and rules to verify the validation of items in a document. It is possible to specify data types of elements in an ontology with XSD.

2.3. Renarration

Renarration is the process of forming an alternative version for a pre-existing narrated web document to make it available for alternative audiences [8] [49]. During the renarration process, DOM (Document Object Model) [50] is dynamically restructured. Therefore, source document that is subject to renarration changes in the client-side and becomes renarrated document.

2.3.1. Semantic Accessibility

Accessibility is the practice of supporting the inclusion of people regardless of their abilities or conditions. In this work, accessibility barriers are divided into two categories, and categorized in terms of the barriers being physical or semantic. Types of barriers that may lead to physical accessibility can be environment, equipment, disability, etc.; semantic accessibility can be literacy, language, culture, expertise etc. The underlying motivation of creating a renarration is to reach alternative audiences: thus, it supports web accessibility mainly in the sense of semantic accessibility through enabling the translation, correction, simplification, elaboration, alteration of a web page with text, audio, image, video content [51] [52] [9].

3. RELATED WORK

This chapter contains the related works and their comparison with this work. The related works chapter is divided into two main sections, because of the significant similarities both renarration and annotation processes share. Annotation works and renarration works sections are in the Section 3.2 and the Section 3.3, respectively.

3.1. Comparison of Renarration and Annotation

To compare renarrations and annotations in a better way, a definition for these processes is necessary. Annotation is the process of linking pieces of information and it allows associating additional information to target selection on a web document. Renarration is the process of creating an alternative version for a pre-existing web document to reach people with diverse backgrounds.

Significant similarities between renarrations and annotations can be listed as follows: they both may support semantic accessibility by giving additional information about the web resource; keep provenance by storing the origin of the web resource; don't interfere or change the original web resource on the server-side; actualize changes on the client-side.

Despite their similarities, renarrations and annotations have several differences. Renarrations target the whole web resource, whereas annotations target part of the web resource, and in this sense annotations are similar to renarration transformations. Renarrations may restructure source documents extensively by replacing, removing, or inserting HTML elements on the client-side, whereas annotations may just attach additional pieces of content on the client-side. One of the core components of renarrations, namely renarration transformations, are rigorously ordered and connected and they need to be applied to DOM in an order, whereas annotations generally are not.

3.2. Annotation Works

3.2.1. Hypothesis

Hypothesis [53] [54] is an open source effort that enables creating annotations on the various resources, e.g. web page, pdf. Users can view and create annotations on resources with using browser extension or proxy of Hypothesis, and owners can embed it in their web pages to allow viewers to annotate these web pages [55]. Hypothesis enables taking notes on resources and sharing them with others. It uses the annotation standards created by Web Annotation Working Group [56] as an infrastructure. The approach of using the browser extension in this work is similar to Hypothesis' use of browser extension; however, the use of renarrations in the prototype, instead of annotations, to improve semantic accessibility of content and to reach alternative audiences creates a significant difference.

3.2.2. Genius Web Annotator

Genius Web Annotator [57] is an annotation service of "Genius.com". Users can view and create annotations on web pages by using proxy of Genius, and publishers can embed the javascript code of Genius to their web pages for having the annotation functionality. The prototype in this work diverges from the Genius with the use of browser extension, decentralized social linked data project Solid [17] and renarrations.

3.2.3. Annotator

Annotator [58] is an open-source JavaScript library that enables adding annotation functionality to web pages for building annotation applications. It provides tools for annotating content in web pages. Annotator library mainly serves annotation functionality to the web pages in the development process, while the prototype in this work serves to the end-user to enable them to renarrate web pages.

3.2.4. Dokieli

Dokieli [59] [60] is an open source client-side editor that enables decentralized article publishing, annotations and social interactions via using several recommendations of W3C such as Web Annotation Data Model [13], Linked Data Platform [61], Linked Data Notifications [62], Activity Streams. It uses HTML+RDFa to enable people to annotate, edit and publish machine-readable documents that can be stored on a decentralized Linked Data Platform [61] e.g., Solid [63]. It implements Web Annotation Data Model [13] to enable individuals to create and view annotations, and Linked Data Notifications [62] to notify people about social activities and annotations. Dokieli's approaches of using social interaction elements, being able to edit a web document and enabling the storage of the semantic data in a decentralized manner are similar to the ones in this work. Dokieli uses HTML+RDFa based method to edit and save documents, whereas the prototype in this work uses a renarration ontology to represent the changes in the source document (renarration) and doesn't store the parts of the original source document in a renarration turtle file.

3.3. Renarration Works

3.3.1. Alipi

Alipi [8] is a framework for supporting accessibility over the Web by using renarrations. Renarration is an alternative rendition of a web page; it has the aim of reaching alternative audiences with varied accessibility problems due to various reasons: e.g., language, age, geographic location. The Alipi web application uses external meta-data for representing renarrations, whereas the prototype, RNEx, in this work describes renarrations with a renarration ontology that represents the creation of renarrations and social interactions. Furthermore, RNEx uses Solid [17] to realize the social and decentralized manner of renarration.

3.3.2. Sweet

Sweet [9] is a framework aiming to overcome the new accessibility challenges, i.e., linguistic, socio-cultural, cognitive type of barriers using the technique of renarration. Sweets are external meta-data used in web page transformation, renarration, process and they are composed of UserID, Context, Resource, Attributes, and TimeStamp elements. Sweet represents renarrations with external metadata, whereas in this work creation of renarrations and social interactions are represented with a renarration ontology.

3.3.3. Renarration Data Model

Renarration Data Model [11] [10] is an ontology that represents the fundamental renarration elements, e.g. renarration, renarration transformation, document, renarrator. While both the ontology in this work, Renarration Social Ontology, and Renarration Data Model [10] use main components of renarration e.g., renarrator, document, renarration transformation, selector classes, Renarration Social Ontology diverges from Renarration Data Model with the use of social interactions elements. Furthermore, Renarration Social Ontology reuses the existing ontologies, Web Annotation Data Model [13] for representing selectors, and the Activity Streams 2.0 [14] for activities and social interactions, e.g. creation of a renarration, following a renarrator. Lastly, the way of utilizing common main components and corresponding classes, object properties, data properties, and their relations differ from one ontology to the other.

4. REQUIREMENTS

This chapter describes the requirements of a social renarration platform. The ontology described in Chapter 5 and the prototype presented in Chapter 6 adhere to these requirements. Initially we introduce some terms that are used in the requirements as well as the remainder of this thesis. Thereafter, the requirements are provided.

4.1. Glossary

The following terms are used throughout this thesis and within the requirements. The terms are listed in alphabetical order. The glossary terms that are used in defining another term are italicized.

Action: An operation that describes the nature of the transformation related to a renarration transformation. It is specifically related to the *selection* in the source document. We focus on three types of action: "replace", "remove" or "insert".

Activity Streams 2.0: A W3C recommendation that provides a machine-readable and extensible specification that describes common activities on social platforms [14].

Document: A Web document.

Document Object Model (DOM): A tree structured specification of a web document. It is a representation that lends itself to automated processing.

Internationalized Resource Identifiers (IRI): A unique resource identifier that allows the use a great variety of character codes. In this work IRIs are used to refer to *documents*, *renarrations*, and *renarrators*.

Narration: A multimedia expression ("text", "image", "audio" or "video") provided by a person to describe something.

In the context of this work, it is related to an alternative narration related to a renarration transformation.

Renarration: An alternative version of a web resource by provided by a renarrator via transformations such as elaboration, simplification, and contextualization. It aims to render a resources accessible to another audience, such as one that speaks another language or is insufficiently unfamiliar with the context of the original source. A renarration consists of a set of *renarration transformations* that are provided by a renarrator on a given date and a specific *document*.

RNEx: A browser extension that enables the creation of renarrations with social interactions.

Renarration Social Ontology: An ontology introduced in this thesis that specifies the concepts and properties of renarrations with social interactions.

Motivation: A reason that declares why a renarration is provided such as "correction", "simplification", "translation", "alteration", and "elaboration".

Renarrator: A person who provides a renarration. In this thesis a renarrator is identified via their *WebID*.

Renarration Transformation: A description of how to transform a part of the source *document*, which may include omitting it or an alternative *narration*.

Response: An interaction with a renarration, such as rating and commenting.

Selector: A specification of a location within a web *document*.

Solid Project: A re-decentralization of web project that separates the data and functionality of applications in a way provide privacy and control of access of the data to its owner [17].

Source Document: A document that is being subjected to a renarration.

XPath: A language for identifying nodes within HTML/XML documents [64].

Web Annotation Data Model: A W3C recommendation that specifies web scale annotations [13].

Browser extension: A software program that delivers its functionality through web browsers [15].

WebID: A unique identifier (via IRI) for agents such as persons and organizations. A WebID Profile (Profile Document) provides information about an agent and WebID specifies the agent. In this work, in accordance with the Solid architecture, it is used to describe people and manage their access rights.

4.2. Functional and Non-Functional Requirements

In the following requirements the decentralization functionality is defined based on the Solid decentralized web project [17].

(i) Functional Requirements
  (i) Authentication
    (i) Users shall be able to login with a WebID obtained from the decentralized web project Solid.
  (ii) Authorization
    (i) Users shall be able to view the renarrations which they are authorized to access.
    (ii) Users shall be able to see the responses to renarrations that they are authorized to access.

(iii) Users shall be able to specify the access rights to their renarrations and responses by specifying whom (self, individuals, any one who is logged in, groups, public, software bots, and web apps) and how they can access them (read, write, and control).

(iii) Renarration

(i) Users shall be able to create a renarration by specifying sequence of renarration transformation and motivation

(ii) Renarration Transformations

(ii) Users shall be able to create a renarration transformation by providing: selector, action, narration

(ii) User shall be able to identify an element within an HTML document using a selector

(ii) User shall be able to identify some text or characters within an HTML element using a selector

(ii) User shall be able to specify an action "replace", "remove", "insert" related to a selection.

(ii) User shall be able to provide an alternative narration related to a selection for the "replace" and "insert" action types.

(ii) User shall be able to create narrations with multiple media types i.e. "audio, video, image, text"

(ii) User shall be able to specify language of a narration

(iii) User shall be able to specify one or more motivations for a renarration such as "translation, simplification, correction, alteration, elaboration".

(iv) User shall be able to choose and view a renararration among list of renarrations about visited source document

(v) System shall be able to find XPath of chosen HTML element and apply the change

(vi) System shall get the WebID of the user and IRI of source document to set respectively the renarrator and source document of the renarration

(vii) System shall be able to compound the renarration inputs of user and renarration data from system to create renarration turtle file using the Renarration Social Ontology

(viii) System shall persist a renarration by composing the source document, renarrator Id, creation time (timestamp), renarration transformations and motivations, and store it in the personal online data store (pod) of the renarrator, timestamp the creation of renarration transformation and renarration

(ix) System shall indicate when a user is viewing a renarrated document

(x) System shall be able to apply a renarration to source document by restructuring the DOM when user viewing a renarration

(iv) Social Aspects

(i) Response

(i) User shall be able to rate a renarration on a scale of 1-5 value, 1 being very bad and 5 being very good

(i) User shall be able to respond a renarration with a text comment

(i) User shall be able to view the responses of a renarration

(i) System shall be able to show the responses about the corresponding renarrations on the UI

(ii) User shall be able to follow a renarrator

(iii) System shall recommend renarrations to renarrators

(iv) System shall notify users about the renarrations and responses created by the users they follow

(ii) Non-Functional Requirements

(i) The system shall utilize the Solid Project to handle the decentralization and privacy aspects

(ii) The system shall always maintain the provenance of source documents

(iii) All user generated data will be represented using ontologies including Renarration Social Ontology

(iv) The system shall enable privacy for the renarrations

(v) User shall be able to access all renarration related content in accordance with the Solid Access Control Lists (ACL) specification

(vi) System shall be able to record activities in accordance with the Activity Streams 2.0

(vii) System shall support SPARQL queries

4.3. Representation

Seven steps of Ontology 101 development [65] process for the proposed Renarration Social Ontology can be seen in the below.

(i) *Determine the domain and scope of the ontology:* The proposed ontology represents renarrations, web documents, and social interactions of renarrators and renarrations, and interconnections between them. In other words, the ontology describes the social semantic renarration data of a social renarration platform, e.g., renarrations, renarrators, web documents, social interactions.

(ii) *Consider reusing existing ontologies:* A revised version of Renarration Data Model [10] is used to describe the document, renarrators, renarrations and subcomponents of it. Web Annotation Data Model [13] ontology is used to describe the selectors that are used to indicate the selection parts of renarration transformations. Activity Streams 2.0 [14] ontology is used to represent the activities and social interactions, e.g. creation of a renarration, following a renarrator. In addition to these ontologies, Dublin Core Elements [45], Friend Of A Friend Vocabulary [43] ontologies are used to define various components.

(iii) *Enumerate important terms in the ontology:* Important terms in the ontology are specified to represent the social semantic renarration data of social renarration platforms. These terms include response, comment, rate, renarrator, document, renarration, renarration transformation, narration, selector, action, motivation.

(iv) *Define classes and the class hierarchy:* Classes and class hierarchy are defined for the important enumerated terms of social renarration platforms. In the Figure 4.1 classes and class hierarchy can be seen. For example, the *rnsoc:Response* class has two subclasses *rnsoc:Rate* and *rnsoc:Comment*, similarly *rnsoc:Narration* class has four subclasses, i.e. *rnsoc:Text*, *rnsoc:Image*, *rnsoc:Video*, *rnsoc:Audio*.

(v) *Define the properties of classes and slots:* Object properties and data properties are defined to represent the relations and properties of the ontology. The following are some examples of object and data properties. The *rnsoc:hasResponse* object property has the domain of *rnsoc:Renarration* class and range of *rnsoc:Response* class, similarly *rnsoc:onSourceDocument* object property has the domain of *rnsoc:Renarration* class and range of *rnsoc:Document* class. The *rnsoc:hasCommentValue* data property has the domain of *rnsoc:Response* class and range of *xsd:string* value, and *rnsoc:hasAction* data property has the domain of *rnsoc:RenarrationTransformation* class and range of "Replace"^^*xsd:string*, "Remove"^^*xsd:string*, "Insert"^^*xsd:string* values.

(vi) *Define the facets of the slots:* Value restrictions and cardinality constraints are defined in the proposed ontology. The following are some examples of value restrictions and cardinality constraints. Likewise, value restriction is used for the *rnsoc:hasMotivation* object property which has the domain of *rnsoc:Renarration* class and range of "Alteration"^^*xsd:string*, "Correction"^^*xsd:string*, "Elaboration"^^*xsd:string*, "Simplification"^^*xsd:string*, "Translation"^^*xsd:string* values. Similarly, cardinality constraint is used for *rnsoc:hasRateValue* data property which has the domain of *rnsoc:Response* class and range of (*rnsoc:hasRateValue* exactly 1 *xsd:integer*) value.

(vii) *Create instances:* Individuals are created using the prototype RNEx and they are stored in Solid pods in the turtle format.

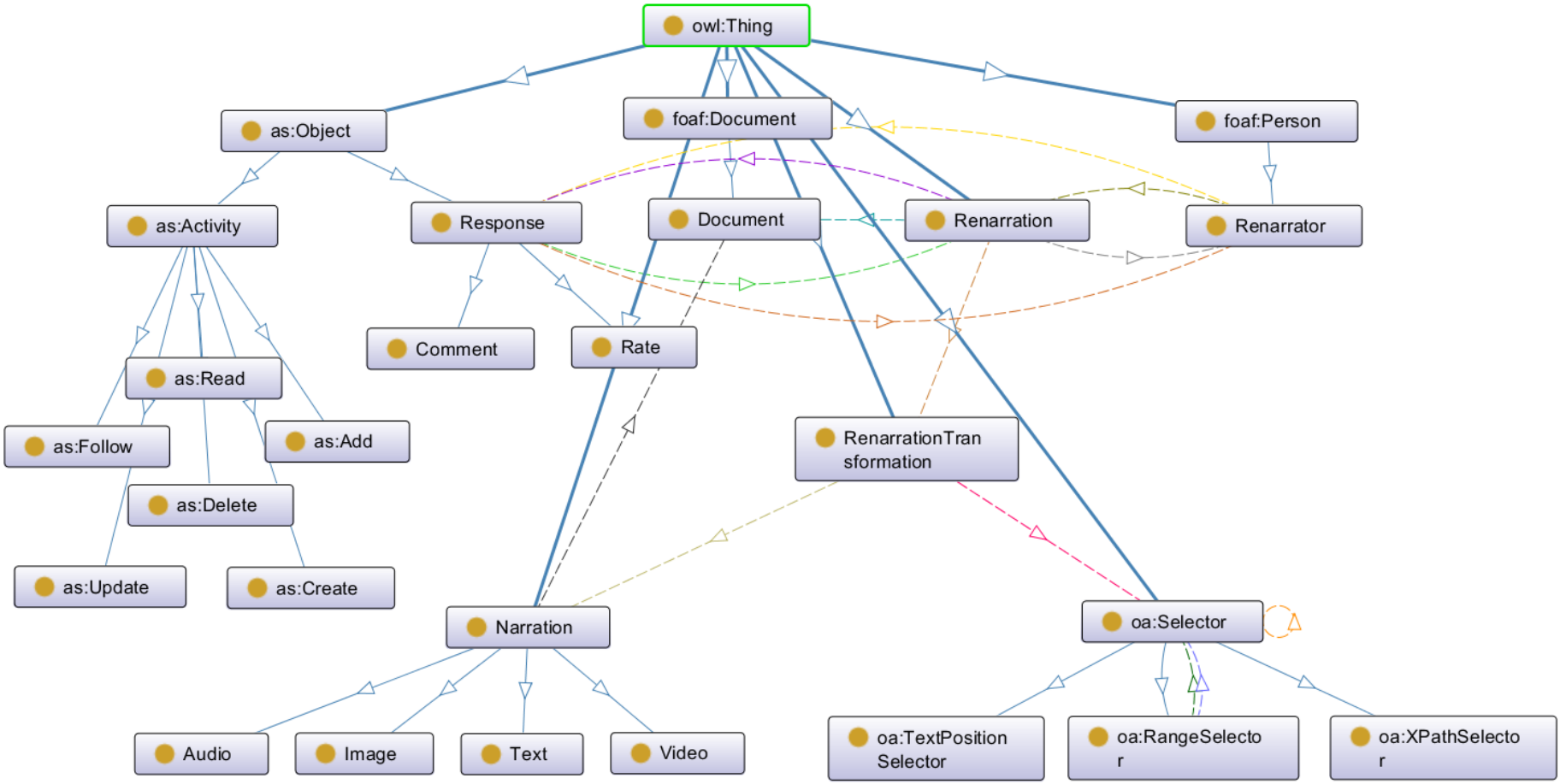

Figure 4.1. Class overview of the Renarration Social Ontology. Solid arrows indicates the subclass relation and dotted arrows remarks specific object properties.

5. MODEL

This chapter, explains the Renarration Social Ontology. The ontology essentially contains seven core elements: Renarration, Document, Renarrator, Renarration Transformation, Selector, Narration, Response. This chapter details the classes, object properties and data properties of this ontology. We provide diagrams and examples for illustration purposes. Instances of renarrations are referred to as renarration specifications. They are always represented using the Turtle syntax (See Turtle Section 2.1.6).

5.1. Renarration Social Ontology

Renarration Social Ontology has been created mainly taking the Renarration Data Model [10], Web Annotation Data Model [13], and Activity Streams [14] ontologies into consideration [66]. Renarration Data Model gives guidance on main renarration components. Annotation Data Model contributes a set of semantic elements related to the selectors for defining the creation of the renarration. Activity Streams ontology contributes to activities and social parts, the interactions of the renarrations and renarrators. In addition to these significant ontologies Dublin Core Elements [45], Friend Of A Friend Vocabulary [43] ontologies have been utilized to define a set of key elements. Consequently, Renarration Social Ontology consults and correlates the ontologies mentioned above as building blocks with additional ontological components to represent renarrations and related social interactions.

The Renarration Social Ontology fundamentally provides semantic definitions for representing the creation of renarrations and certain social interactions between the renarrators and the renarrations. Table 5.1 shows the prefixes of Renarration Social Ontology and utilized ontologies during the creation of it.

Table 5.1. Renarration Social Ontology And Reused Existing Ontologies And Their Prefixes.

Prefix	Namespace	Description
rnsoc	http://soslab.cmpe.boun.edu.tr/ontologies/rnsoc.owl	Renarration Social Ontology
oa	http://www.w3.org/ns/oa#	The Web Annotation Data Model [13]
dc	http://purl.org/dc/elements/1.1/	Dublin Core Elements [45]
rdf	http://www.w3.org/1999/02/22-rdf-syntax-ns#	Resource Description Framework [67]
owl	http://www.w3.org/2002/07/owl#	Web Ontology Language [68]
xsd	http://www.w3.org/2001/XMLSchema#	XML Schema Definition [69]
foaf	http://xmlns.com/foaf/0.1/	Friend Of A Friend Vocabulary [43]

The Figure 5.1 shows the visualization of the ontology. Ellipses represent classes. Rounded squares represent data type of literals. Arrows pointing to ellipses represent the object properties. Arrows pointing to the rounded squares represent the data properties. Dotted arrows represent the *rdfs:subClassOf* relation, hence, head of the arrow points the superclass. To ease the explanation of the Renarration Social Ontology, the whole graph of the ontology, Figure 5.1, has been separated into partial graphs in each subsection. These subsections contain the classes of the ontology and their significant relations.

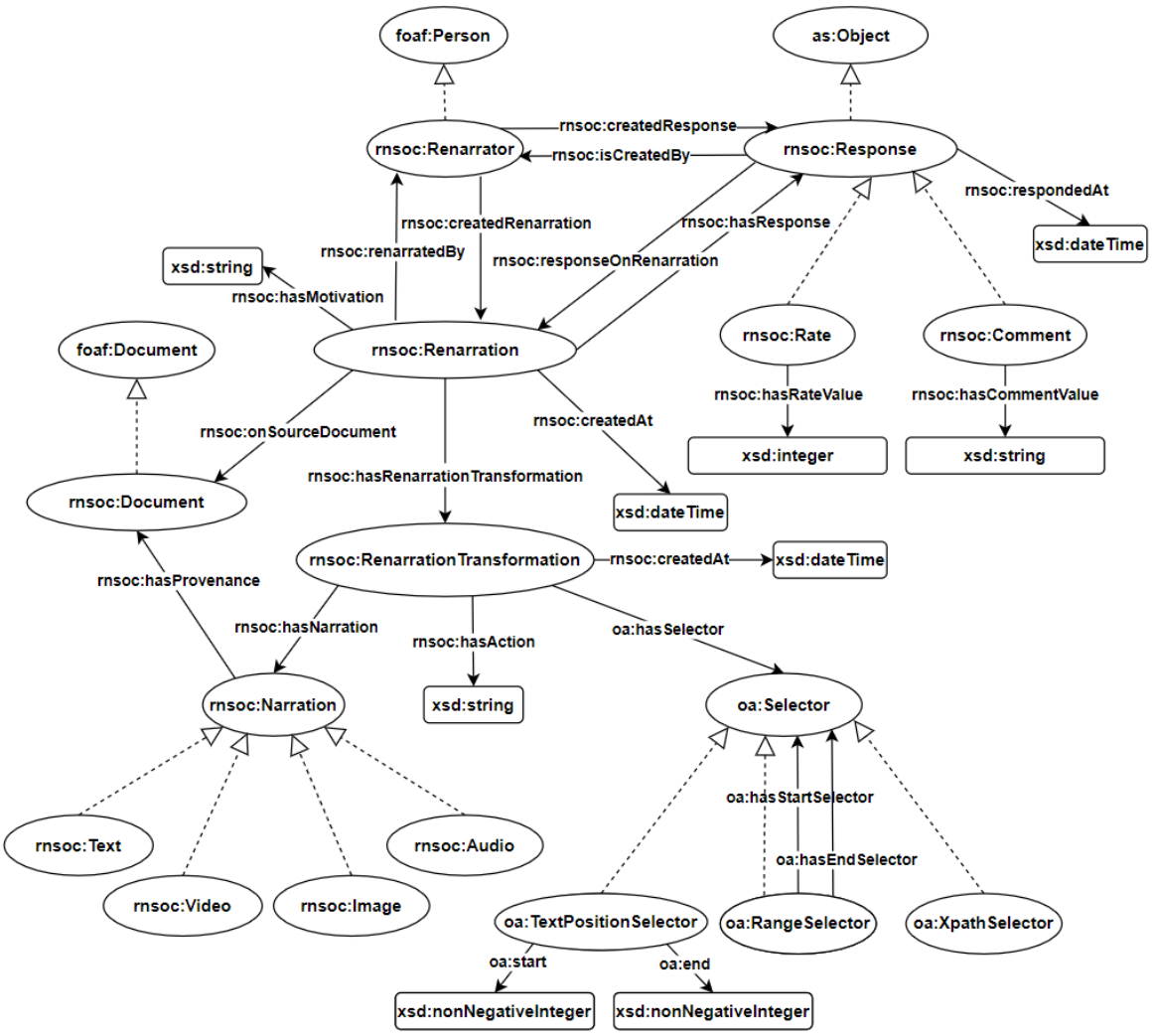

Figure 5.1. Renarration Social Ontology

5.1.1. Renarration

Renarration is the process of composing an alternative for a pre-existing web document to reach alternative audiences. Technically it is the process of creating a rendition for a pre-established web document, with changing it on the client-side, via structuring DOM (Document Object Model), to enhance the semantic accessibility of source web document with the motivations such as "elaboration, translation, simplification, alteration, correction". Fundamentally, renarration consists of a collection of renarration transformations and semantic elements such as document, renarrator, motivation, creation time.

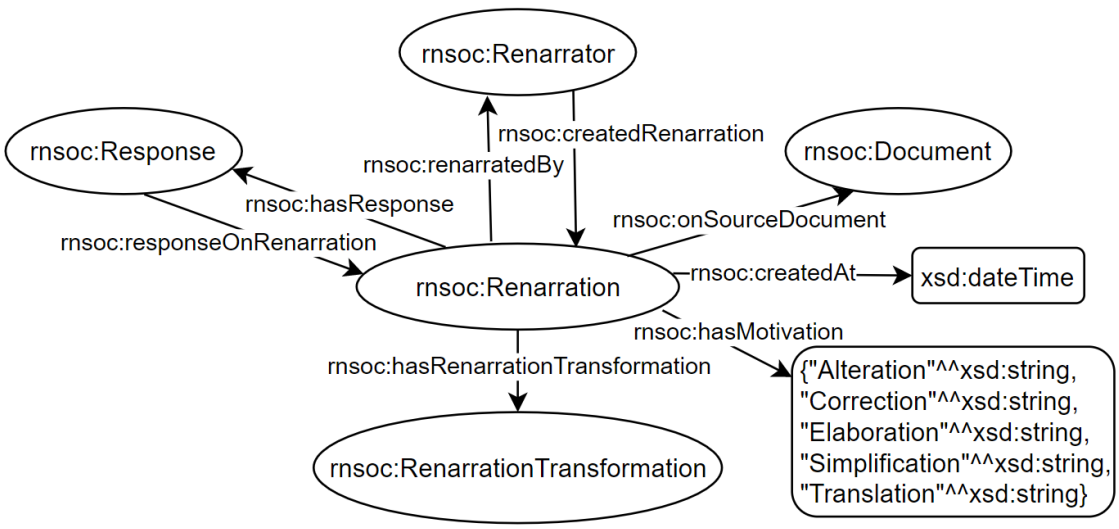

Figure 5.2. Renarration class and its significant relations

In the Figure 5.2 renarration class and its relations to document, renarrator, renarration transformation and response classes, and object properties that describes these relations can be seen. The Table 5.2 shows the renarration class and associated prominent elements.

Table 5.2. Description of renarration class and directly associated elements.

Term	Type	Description
rnsoc:Renarration	Class	Renarration is an alternative narration for a pre-existing web resource with the intent of reaching alternative audiences
rnsoc:hasMotivation	Data Property	Specifies the motivation of the renarration. The value of it should be any of these five; "Alteration", "Correction", "Elaboration", "Simplification", "Translation"
rnsoc:createdAt	Data Property	Specifies the creation date and time of renarration or renarration transformation

Example Case: Eren finds a web document about the life of a musician and decides to create a renarration to elaborate it with a concert video.

```
@prefix : <#>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix owl: <http://www.w3.org/2002/07/owl#>.
@prefix rnsoc: <http://soslab.cmpe.boun.edu.tr/ontologies/rnsoc.owl>.
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.
@prefix oa: <http://www.w3.org/ns/oa#>.
@prefix dc: <http://purl.org/dc/elements/1.1/>.
@prefix ren: <https://neua.gitlab.io/renarration-test-cases/>.
@prefix c: <https://eren-therenarrator.example.com/profile/card#>.

:kb2ef2a3-4ed31af3-b3a3ed2e
  a owl:NamedIndividual, rnsoc:Renarration;
  rnsoc:createdAt "2020-11-06T09:12:14.003Z"^^xsd:dateTime;
  rnsoc:hasMotivation "elaboration"^^xsd:string;
  rnsoc:hasRenarrationTransformation
    [
      a rnsoc:RenarrationTransformation;
      oa:hasSelector
        [
          a oa:RangeSelector;
          oa:hasStartSelector
            [
              a oa:XpathSelector;
              rdf:value "/html[1]/body[1]/p[1]"
            ];
          oa:hasEndSelector
            [
              a oa:XpathSelector;
              rdf:value "/html[1]/body[1]/p[2]"
            ]
        ];
      rnsoc:createdAt "2020-11-06T09:05:11.001Z"^^xsd:dateTime;
      rnsoc:hasAction "insert"^^xsd:string;
      rnsoc:hasNarration
        [
          a rnsoc:Video;
          dc:format "text/html";
          dc:language "en";
          rdf:value
            "<video controls>
              <source src="http://example.com/video.mp4" type="
     video/mp4">
             </video>"
        ]
    ];
  rnsoc:onSourceDocument ren:renarration_example;
  rnsoc:renarratedBy c:me.
ren:renarration_example a rnsoc:Document.
```

Figure 5.3. A renarration specification that elaborates the source document by inserting a video

Figure 5.3 shows Eren's renarration specification. The address of the source document is specified with the *rnsoc:onSourceDocument* object property whose value is

ren:renarration_ example (Line:47). Eren's motivation is specified with *rnsoc:hasMotivation* data property with value "elaboration" (Line:14). The creation date is specified with the *rnsoc:createdAt* data property (Line:13). The renarration transformation to insert a video is specified with an instance of the *rnsoc:RenarrationTransformation* class (Line:16 - Line:44). The instances of the *rnsoc:Renarration* (Line:12) and *rnsoc:RenarrationTransformation* (Line:17) classes are related with the *rnsoc:hasRenarrationTransformation* object property (Line:15).

5.1.2. Renarration Transformation

Renarration transformation is a description of how to transform a part of the source document. Therefore, a renarration may contain one or more renarration transformations depending on the number of modifications made on a source document. In the Figure 5.4 renarration transformation class and its significant relations can be seen.

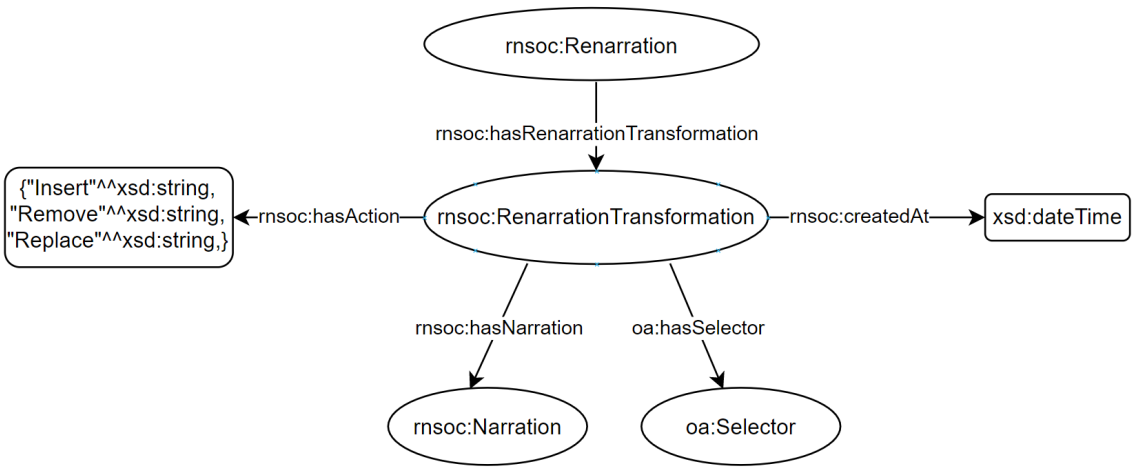

Figure 5.4. RenarrationTransformation class and its significant relations

An instance of the *rnsoc:RenarrationTransformation* class should have a *rnsoc:hasAction* data property with the value of any of these three "Replace", "Remove", "Insert". An instance of the *rnsoc:RenarrationTransformation* class should have a *oa:hasSelector* object property. The *oa:hasSelector* object property has a domain of *rnsoc:RenarrationTransformation* class and range of the *oa:Selector* class. An instance of the

rnsoc:RenarrationTransformation class might have a *oa:hasNarration* object property depending on a type of action. The *oa:hasNarration* object property has a domain of *rnsoc:RenarrationTransformation* class and range of the *rnsoc:Narration* class. The Table 5.3 shows the renarration transformation class and associated prominent elements.

An instance of the *rnsoc:RenarrationTransformation* class should have a *rnsoc:hasAction* data property that may have one of the following values "remove", "replace" or "insert". The remove action indicates the removal of the fragment specified by a selector (*rnsoc:hasSelector*). The replace action indicates the replacement of the fragment specified by a selector with an alternative narration specified with the *rnsoc:hasNarration* object property.

```
oa:hasSelector
 [
  a oa:RangeSelector;
  oa:hasStartSelector
    [
      a oa:XPathSelector;
      rdf:value ""
    ];
  oa:hasEndSelector
    [
      a oa:XPathSelector;
      rdf:value "/html[1]/body[1]/p[1]"
    ]
];
```

Figure 5.5. A selector that specifies an insertion point to the beginning of the document

The insert action is more complicated since it is difficult to specify to the insertion point. HTML documents consist of a nested sequence elements. This presents a problem when we wish to express an insertion at the beginning or at the end of a sequence. If we try to indicate an insertion to be after an element, we can not specify an insertion at the beginning of a sequence. Alternatively if we try to indicate an insertion point to be before an element, we can not specify an insertion to the end of a sequence.

To address this issue we specify insertion points as two consecutive selectors: *rnsoc:hasStartSelector* and *rnsoc:hasEndSelector*. The former indicates the previous element and the latter indicates the succeeding element. Furthermore, we introduce the null address (empty string) to indicate the absence of an element. When the *rnsoc: hasStartSelector* has a null address it indicates the very beginning of the sequence. Similarly, when the *rnsoc:hasEndSelector* has a null address it indicates the end of the sequence. Figure 5.5 shows a selector that points to very beginning of an HTML document.

Table 5.3. Description of renarration transformation class and directly associated elements.

Term	Type	Description
rnsoc:RenarrationTransformation	Class	Renarration transformation is a subcomponent of a renarration that represents the renarrator-generated atomic changes on a web resource
rnsoc:hasRenarrationTransformation	Object Property	Indicates the renarration transformation that is the part of renarration
oa:hasSelector	Object Property	Focused fragment of the source resource [13]
rnsoc:hasAction	Data Property	Specifies the action of the renarration transformation. The value of it should be any of these three; "Replace", "Remove", "Insert"
rnsoc:hasNarration	Object Property	Indicates the narration that belongs to a renarration transformation
rnsoc:createdAt	Data Property	Specifies the creation date and time of renarration or renarration transformation

Example Case: Mary creates a renarration that contains two renarration transformations. The first renarration transformation replaces a heading element with another heading element and the second renarration transformation removes a paragraph element.

```
@prefix : <#>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix owl: <http://www.w3.org/2002/07/owl#>.
@prefix rnsoc: <http://soslab.cmpe.boun.edu.tr/ontologies/rnsoc.owl>.
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.
@prefix oa: <http://www.w3.org/ns/oa#>.
@prefix dc: <http://purl.org/dc/elements/1.1/>.
@prefix ren: <https://neua.gitlab.io/renarration-test-cases/>.
@prefix c: <https://mary-therenarrator.example.com/profile/card#>.

:keblyc2x-3ac1d1el-gdf6a35e
  a owl:NamedIndividual, rnsoc:Renarration;
  rnsoc:createdAt "2020-10-12T14:13:23.005Z"^^xsd:dateTime;
  rnsoc:hasMotivation "alteration"^^xsd:string;
  rnsoc:hasRenarrationTransformation
    [
      a rnsoc:RenarrationTransformation;
      oa:hasSelector
        [
          a oa:XPathSelector;
          rdf:value "/html[1]/body[1]/h1[1]";
        ];
      rnsoc:createdAt "2020-10-12T14:05:21.005Z"^^xsd:dateTime;
      rnsoc:hasAction "replace"^^xsd:string;
      rnsoc:hasNarration
        [
          a rnsoc:Text;
          dc:format "text/html";
          dc:language "en";
          rdf:value "<h1>Some Text</h1>"
        ]
    ];
    [
      a rnsoc:RenarrationTransformation;
      oa:hasSelector
        [
          a oa:XPathSelector;
          rdf:value "/html[1]/body[1]/p[2]";
        ];
      rnsoc:createdAt "2020-10-12T14:08:18.005Z"^^xsd:dateTime;
      rnsoc:hasAction "remove"^^xsd:string;
    ];
  rnsoc:onSourceDocument ren:renarration_transformation_example;
  rnsoc:renarratedBy c:me.
ren:renarration_transformation_example a rnsoc:Document.
```

Figure 5.6. A renarration specification with two renarration transformations

Figure 5.6 shows the renarration specification for the action example case. The first *rnsoc:RenarrationTransformation* that Mary created is indicated with the statement in the Line:17. *oa:hasSelector* (Line:18) object property indicates that the blank node between the Line:19 and Line:22 is a *oa:Selector* class instance. Action of the first *rnsoc:RenarrationTransformation* (Line:17) that has the intent of replacing heading with another heading can be seen in the Line:24, hence it has the value of "Replace". *rnsoc:hasNarration* (Line:25) object property indicates that the blank node between the Line:26 and Line:31 is a *rnsoc:Narration* class instance.

The second *rnsoc:RenarrationTransformation* can be seen between the Line:33 and the Line:42. *oa:hasSelector* (Line:35) object property indicates that the blank node between the Line:36 and Line:39 is a *oa:Selector* class instance. Action of the second *rnsoc:RenarrationTransformation* (Line:34) that has the intent of removing the paragraph element can be seen in the Line:41, thus it has the value of "Remove".

5.1.3. Selector

Selector is the segment that is chosen for pointing out the part of the source document to apply a renarration transformation. Each *rnsoc:RenarrationTransformation* should have one type of *oa:Selector* and the relation of two should be defined with the *oa:hasSelector*. In the Figure 5.7 selector class and its relation to renarration and renarration transformation classes, and its subclasses can be seen.

Instead of creating a custom selector class, Web Annotation Data Model [13] *oa:Selector* class was used, because it is a well-thought-out class and its sub-classes were created by considering various cases in detail and Web Annotation Working Group [56] made extensive work on it covering the scenarios related to selecting XPath, text, SVG, etc. The Table 5.4 shows the selector class and associated prominent elements.

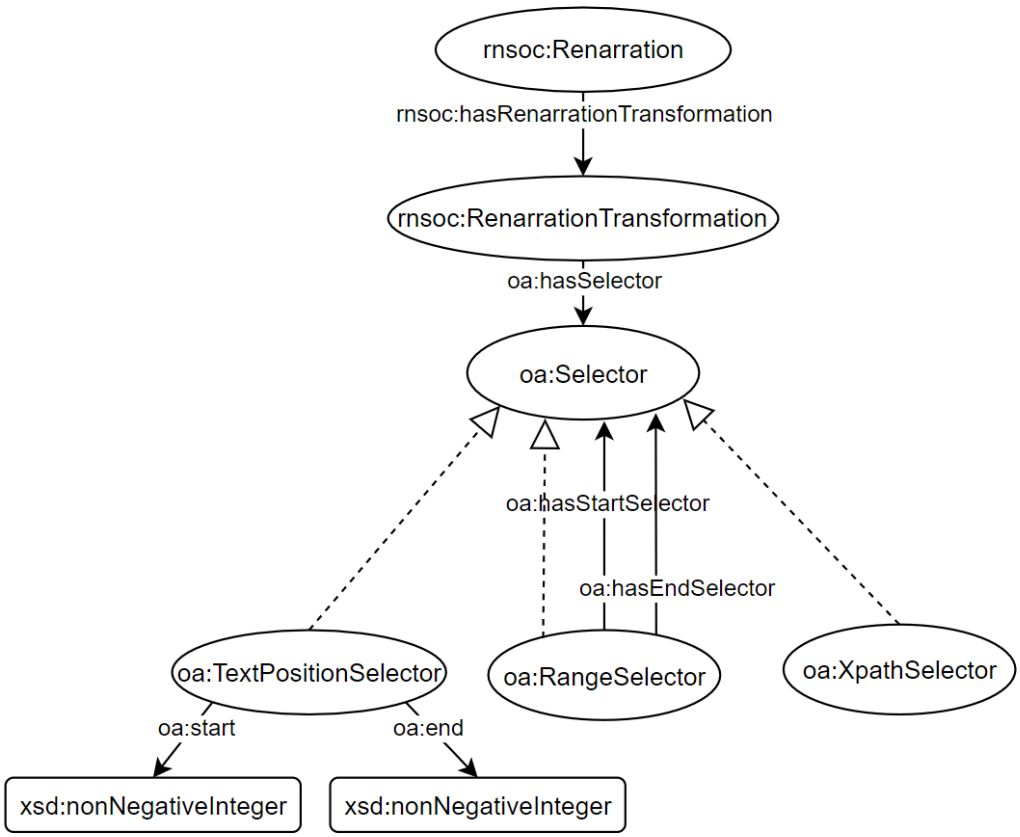

Figure 5.7. Selector class and its significant relations

Table 5.4. Description of selector class and directly associated elements.

Term	Type	Description
oa:Selector	Class	Focused fragment of the source resource [13]
oa:start	Data Property	Describes the start position of focused content [13]
oa:end	Data Property	Describes the end position of focused content [13]
rdf:value	Data Property	Specifies the structured value [24]
oa:XpathSelector	Class	Specifies a Xpath value for the resource that supports DOM [13]
oa:RangeSelector	Class	Identifies the start and the end of the selection with other Selectors [13]
oa:hasStartSelector	Object Property	Describes the start of the Range Selector with a Selector [13]
oa:hasEndSelector	Object Property	Describes the end of the Range Selector with a Selector [13]
oa:TextPositionSelector	Class	Specifies a text selection with start and end positions [13]

Example Case: John sees an uninteresting web resource with plain text and decides to create a renarration to make it more interesting. Thus, he selects a paragraph element, inserts an image before it and concludes his first renarration transformation, and chooses another paragraph and removes an unnecessary word in it and completes his second renarration transformation, then finishes his renarration.

Figure 5.8 shows the renarration specification for the selector example case. The blank node is in the Line:17 is the first *rnsoc:RenarrationTransformation*, that John created.

Line:18 shows the *oa:hasSelector* object property that connects the *rnsoc:RenarrationTransformation* (Line:17) and *oa:RangeSelector* (Line:20) blank nodes.

oa:hasStartSelector (Line:21) shows the start paragraph element and *oa:hasEndSelector* (Line:26) shows the end paragraph element. *oa:XpathSelector* (Line:23) information indicates that the value in the *rdf:value* (Line:24) is selected element. Consequently, John inserted an image between these paragraph elements.

The blank node is in the Line:44 is the second *rnsoc:RenarrationTransformation*, that John created. The second renarration transformation contains *oa:TextPositionSelector* (Line:51), because John selected a text in a paragraph and removed it.

The start position of the selected text is described with the *oa:start* (Line:52) property and the end position of the selected text is described with the *oa:end* (Line:53) property, eventually, John removed a word from a paragraph element.

```
@prefix : <#>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix owl: <http://www.w3.org/2002/07/owl#>.
@prefix rnsoc: <http://soslab.cmpe.boun.edu.tr/ontologies/rnsoc.owl>.
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.
@prefix oa: <http://www.w3.org/ns/oa#>.
@prefix dc: <http://purl.org/dc/elements/1.1/>.
@prefix ren: <https://neua.gitlab.io/renarration-test-cases/>.
@prefix c: <https://john-therenarrator.example.com/profile/card#>.

:kablyg1x-4uc2c1dl-edb6a26d
  a owl:NamedIndividual, rnsoc:Renarration;
  rnsoc:createdAt "2020-10-14T21:27:52.005Z"^^xsd:dateTime;
  rnsoc:hasMotivation "elaboration"^^xsd:string;
  rnsoc:hasRenarrationTransformation
    [
      a rnsoc:RenarrationTransformation;
      oa:hasSelector
        [
          a oa:RangeSelector;
          oa:hasStartSelector
            [
              a oa:XpathSelector;
              rdf:value "/html[1]/body[1]/p[1]"
            ];
          oa:hasEndSelector
            [
              a oa:XpathSelector;
              rdf:value "/html[1]/body[1]/p[2]"
            ]
        ];
      rnsoc:createdAt "2020-10-14T21:12:52.005Z"^^xsd:dateTime;
      rnsoc:hasAction "insert"^^xsd:string;
      rnsoc:hasNarration
        [
          a rnsoc:Image;
          dc:format "text/html";
          dc:language "en";
          rdf:value
           "<img  src="http://example.com/image.png" >"
        ]
    ];
    [
      a rnsoc:RenarrationTransformation;
      oa:hasSelector
        [
          a oa:XPathSelector;
          rdf:value "/html[1]/body[1]/p[3]";
          oa:refinedBy
            [
              a oa:TextPositionSelector;
              oa:start "9";
              oa:end "13";
            ];
        ];
      rnsoc:createdAt "2020-10-14T21:17:52.005Z"^^xsd:dateTime;
      rnsoc:hasAction "remove"^^xsd:string;
    ];
  rnsoc:onSourceDocument ren:selector_example;
  rnsoc:renarratedBy c:me.
ren:selector_example a rnsoc:Document.
```

Figure 5.8. A renarration specification that uses two different selectors for different kinds of transformations (insert, remove)

5.1.4. Narration

Narration is the content of the renarration transformation. In other words, it is the expression of the desired change in the chosen segment on the source document. Narration is represented as a class and *rnsoc:Narration* class has four subclasses, i.e. *rnsoc:Text*, *rnsoc:Audio*, *rnsoc:Video*, *rnsoc:Image*. These subclasses specify content type of a renarration transformation. In the Figure 5.9 narration class, its subclasses and its relation to renarration and renarration transformation classes can be seen.

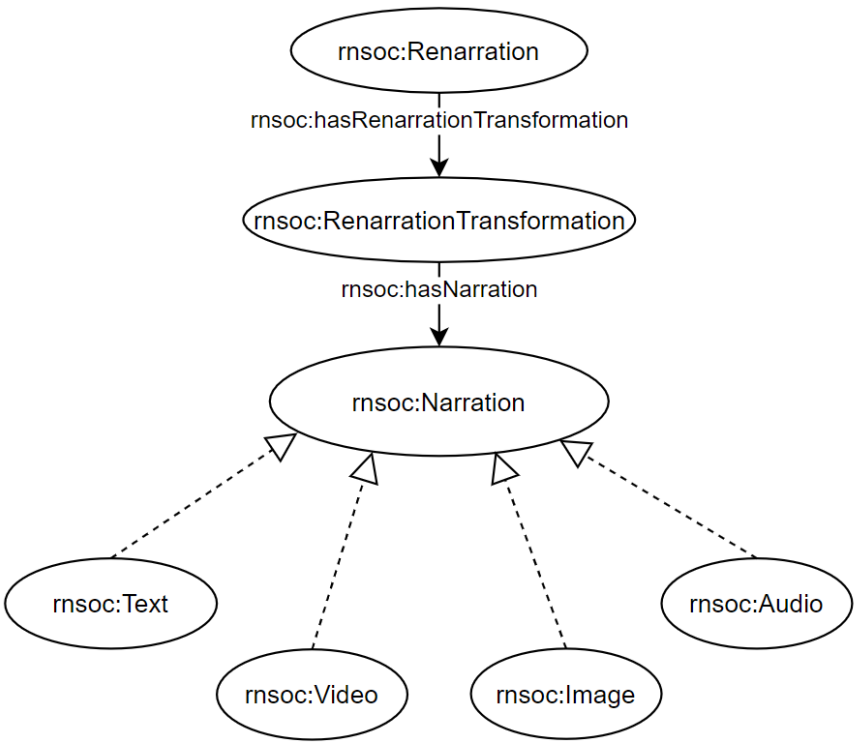

Figure 5.9. Narration class and its significant relations

rnsoc:Narration is the component that indicates the content of *rnsoc:RenarrationTransformation*, hence *rnsoc:RenarrationTransformation* class has the relation of *rnsoc:hasNarration* with the *rnsoc:Narration* class. The Table 5.5 shows the narration class and associated prominent elements.

Table 5.5. Description of narration class and directly associated elements.

Term	Type	Description
rnsoc:Narration	Class	Narration is a renarrator created content that is a subcomponent of a renarration transformation
rnsoc:Audio	Class	Narration with the audio content type
rnsoc:Image	Class	Narration with the image content type
rnsoc:Text	Class	Narration with the text content type
rnsoc:Video	Class	Narration with the video content type
dc:format	Data Property	Specifies the format of the resource and MIME types could be used for the files [45]
rdf:value	Data Property	Specifies the structured value [24]
dc:language	Data Property	Specifies a language value [45]

Example Case: Laura sees a web resource with a literature content and decides to elaborate it with an audio narration. She replaces a long paragraph with an audio element, and concludes the renarration.

```
1 @prefix : <#>.
2 @prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
3 @prefix owl: <http://www.w3.org/2002/07/owl#>.
4 @prefix rnsoc: <http://soslab.cmpe.boun.edu.tr/ontologies/rnsoc.owl>.
5 @prefix xsd: <http://www.w3.org/2001/XMLSchema#>.
6 @prefix oa: <http://www.w3.org/ns/oa#>.
7 @prefix dc: <http://purl.org/dc/elements/1.1/>.
8 @prefix ren: <https://neua.gitlab.io/renarration-test-cases/>.
9 @prefix c: <https://laura-therenarrator.example.com/profile/card
    #>.
10
11 :kecaye2x-3bc1e1de-ede2d21d
12   a owl:NamedIndividual, rnsoc:Renarration;
13   rnsoc:createdAt "2020-11-02T08:34:11.001Z"^^xsd:dateTime;
14   rnsoc:hasMotivation "elaboration"^^xsd:string;
15   rnsoc:hasRenarrationTransformation
16     [
17       a rnsoc:RenarrationTransformation;
18       oa:hasSelector
19         [
20           a oa:XPathSelector;
21           rdf:value "/html[1]/body[1]/p[1]"
22         ];
23       rnsoc:createdAt "2020-11-02T08:14:11.002Z"^^xsd:dateTime;
24       rnsoc:hasAction "replace"^^xsd:string;
25       rnsoc:hasNarration
26         [
27           a rnsoc:Audio;
28           dc:format "text/html";
29           dc:language "en";
30           rdf:value
31             "<audio controls>
32                <source src="http://example.com/audio.mp3" type="
     audio/mpeg">
33              </audio>"
34         ]
35     ];
36   rnsoc:onSourceDocument ren:narration_example;
37   rnsoc:renarratedBy c:me.
38 ren:narration_example a rnsoc:Document.
```

Figure 5.10. A renarration specification that specifies an audio narration to replace a textual paragraph

Figure 5.10 shows the renarration specification for the selector example case. Laura's renarration has only one *rnsoc:RenarrationTransformation* (Line:15). The *rnsoc:hasNarration* object property (Line:25) indicates that *rnsoc:Narration* instance between the Line:26 and Line:34 belongs to *rnsoc:RenarrationTransformation* (Line:15). The *rnsoc:Audio* (Line:27) subclass indicates that the *rnsoc:Narration* instance is an

Audio, *dc:format* of it is "text/html" (Line:28) and *dc:language* of it is "en" (Line:28). These elements express that the language of the audio narration is English. Lastly, *rdf:value* of the narration is an audio element and it can be seen between the Line:31 Line:33. Consequently, the example shows that Laura replaces a paragraph with an audio element.

5.1.5. Response

Response is a way of criticizing or giving an opinion on a renarration, thereby it enables social interactions of the renarrators. In the Figure 5.11 response class, its subclasses, its relation to renarration and renarrator classes, and data and object properties can be seen.

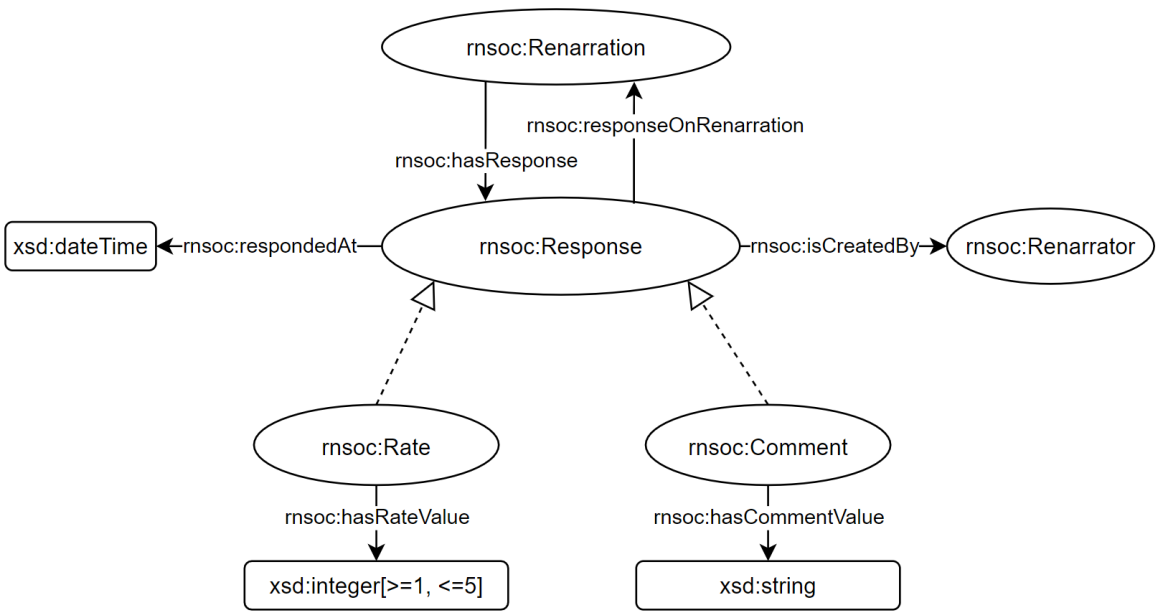

Figure 5.11. Response class and its significant relations

rnsoc:Response is designed as a subclass of *as:Object* so that it enables connection to Activity Streams [14] for further use. *rnsoc:hasResponse* object property connects *rnsoc:Response* class to the *rnsoc:Renarration* class. *rnsoc:Response* has two subclasses *rnsoc:Comment* and *rnsoc:Rate*, the first one has a data property of *rnsoc:hasCommentValue*, and the latter one has a data property of *rnsoc:hasRateValue* for storing the value of comment and rate. The *rnsoc:responseOnRenarration* object prop-

erty indicates the renarration that the response is given to and it is the inverse of *rnsoc:hasResponse* object property that has the domain of *rnsoc:Renarration* and range of *rnsoc:Response*. The Table 5.6 shows the response class and associated prominent elements.

Table 5.6. Description of response class and directly associated elements.

Term	Type	Description
rnsoc:Response	Class	Response is renarrator's reply or reaction to a renarration
rnsoc:Comment	Class	Comment is a type of the response that indicates renarrator's explanation or opinion to a renarration
rnsoc:Rate	Class	Rate is a type of response that indicates a renarrator's opinion about the quality of a renarration. It represents whether renarration is good or bad on a number scale.
rnsoc:hasResponse	Object Property	Specifies the response that is about a renarration
rnsoc:responseOnRenarration	Object Property	Specifies the renarration that is indicated by a response
rnsoc:hasCommentValue	Data Property	Specifies the string value of a comment
rnsoc:hasRateValue	Data Property	Specifies the integer value of a rate, it should be a value from 1 to 5
rnsoc:respondedAt	Data Property	Specifies the creation date and time of response

Example Case: Logan sees a renarration of Xavier and would like to appreciate the valuable work via rating and commenting on it.

```
1  @prefix : <#>.
2  @prefix c: <https://logan-therenarrator.example.com/profile/card
     #>.
3  @prefix foaf: <http://xmlns.com/foaf/0.1/>.
4  @prefix rnsoc: <http://soslab.cmpe.boun.edu.tr/ontologies/rnsoc.owl>.
5  @prefix xsd: <http://www.w3.org/2001/XMLSchema#>.
6  @prefix RNC: <https://xavier-therenarrator.example.com/RNCNT-
     ke1bae4a-5d3bae2c-ael2fad3.ttl#>.
7  @prefix owl: <http://www.w3.org/2002/07/owl#>.
8
9  c:me a foaf:Person, rnsoc:Renarrator.
10
11 :kbe1ad2a-d3ed2a3c-a4da2b1e
12   a rnsoc:Rate;
13   rnsoc:hasRateValue "5"^^xsd:int;
14   rnsoc:isCreatedBy c:me;
15   rnsoc:respondedAt "2020-11-17T09:46:21.746Z"^^xsd:dateTime;
16   rnsoc:responseOnRenarration RNC:ke1bae4a-5d3bae2c-ael2fad3.
17
18 :kda13db2-e4da1a3c-3ab2aca1
19   a rnsoc:Comment;
20   rnsoc:hasCommentValue "Some Text"^^xsd:string;
21   rnsoc:isCreatedBy c:me;
22   rnsoc:respondedAt "2020-11-17T09:55:49.196Z"^^xsd:dateTime;
23   rnsoc:responseOnRenarration RNC:ke1bae4a-5d3bae2c-ael2fad3.
24
25 RNC:ke1bae4a-5d3bae2c-ael2fad3 a owl:NamedIndividual, rnsoc:
     Renarration.
26
```

Figure 5.12. A response to a renarrated document by rating and commenting

Figure 5.12 shows the turtle file of the response example case. Logan's *rnsoc:Rate* (Line:12) to the Xavier's *rnsoc:Renarration* that is *RNC:ke1bae4a-5d3bae2c-ael2fad3* (Line:16) can be seen in the rate part of the turtle file between the Line:11 and Line:16. Logan rates the Xavier's renarration with the value of "5", it can be seen in the Line:13. *rnsoc:hasRateValue* data property indicates the integer value of rate. The *rnsoc:isCreatedBy* object property (Line:14) shows the creator of the rate. The *rnsoc: respondedAt*(Line:15) data property indicates the date of the creation of the rate. The *rnsoc:responseOnRenarration* (Line:16) object property denotes the renarration that the rate is given to.

Similarly, Logan's *rnsoc:Comment* (Line:19) to the Xavier's *rnsoc:Renarration* which is *RNC:ke1bae4a-5d3bae2c-ael2fad3* (Line:23) can be seen in the comment part of the turtle file between the Line:18 and Line:23. Logan's comment of "Some Text" to the

Xavier's renarration can be seen in the Line:20. *rnsoc:hasCommentValue* data property indicates the string value of comment. The *rnsoc:isCreatedBy* object property (Line:21) signifies the creator of the comment. The *rnsoc:respondedAt* (Line:22) data property indicates the date of the creation of the comment. The *rnsoc:responseOnRenarration* (Line:23) object property denotes the renarration that the comment is given to.

5.1.6. Renarrator

Renarrator is the creator of the renarration, it is represented as a class in the Renarration Social Ontology. *rnsoc:Renarrator* class has *rnsoc:renarratedBy* relation with *rnsoc:Renarration* class. *rnsoc:renarratedBy* object property denotes the creator of the renarration. Similarly, *rnsoc:Response* class has *rnsoc:isCreatedBy* relation with *rnsoc:Renarrator* class to indicate the creator of the response. *rnsoc:Renarration* is a subclass of *foaf:Person* therefore it may be associated with the *foaf:Person* properties such as *foaf:name* for the further use. Renarrator class and its relation to renarration and response classes, and object properties of this relation can be seen in the Figure 5.13 . The Table 5.7 shows the renarrator class and associated prominent elements.

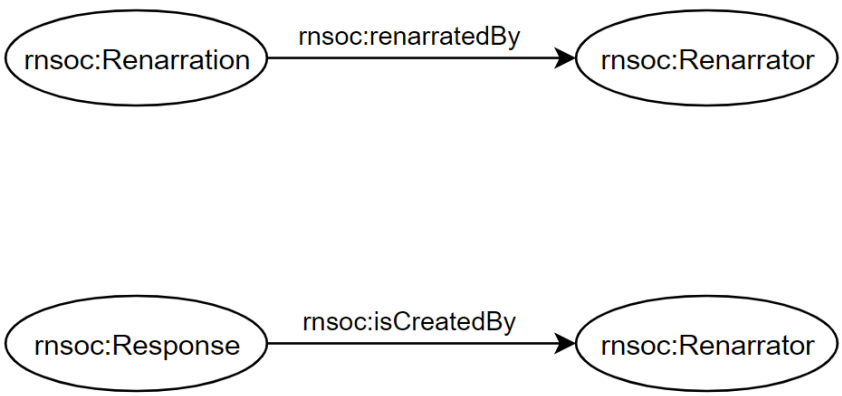

Figure 5.13. Renarrator class and its significant relations

Table 5.7. Description of renarrator class and directly associated elements.

Term	Type	Description
rnsoc:Renarrator	Class	Renarrator is the person who creates renarrations
rnsoc:isCreatedBy	Object Property	Specifies the renarrator who created the response
rnsoc:renarratedBy	Object Property	Specifies the rn:Renarrator who creates the rn:Renarration. At least one rn:renarratedBy property attached to the renarration is necessary

Example Case: Bob sees a redundant and wordy paragraph in a web document and he decides to renarrate the document and removes the redundant paragraph to simplify the document.

```
1 @prefix : <#>.
2 @prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
3 @prefix owl: <http://www.w3.org/2002/07/owl#>.
4 @prefix rnsoc: <http://soslab.cmpe.boun.edu.tr/ontologies/rnsoc.owl>.
5 @prefix xsd: <http://www.w3.org/2001/XMLSchema#>.
6 @prefix oa: <http://www.w3.org/ns/oa#>.
7 @prefix dc: <http://purl.org/dc/elements/1.1/>.
8 @prefix ren: <https://neua.gitlab.io/renarration-test-cases/>.
9 @prefix c: <https://bob-therenarrator.example.com/profile/card#>.
10
11 :kecaye2x-3bc1e1de-ede2d21d
12   a owl:NamedIndividual, rnsoc:Renarration;
13   rnsoc:createdAt "2020-11-04T16:04:12.003Z"^^xsd:dateTime;
14   rnsoc:hasMotivation "simplification"^^xsd:string;
15   rnsoc:hasRenarrationTransformation
16     [
17       a rnsoc:RenarrationTransformation;
18       oa:hasSelector
19         [
20           a oa:XPathSelector;
21           rdf:value "/html[1]/body[1]/div[2]/p[1]"
22         ];
23       rnsoc:createdAt "2020-11-04T16:01:11.006Z"^^xsd:dateTime;
24       rnsoc:hasAction "remove"^^xsd:string;
25     ];
26   rnsoc:onSourceDocument ren:renarrator_example;
27   rnsoc:renarratedBy c:me.
28 ren:renarrator_example a rnsoc:Document.
```

Figure 5.14. A renarration specification that aims to simplify by removing an element

Figure 5.14 shows the renarration specification for the renarrator example case. The *rnsoc:Renarration* (Line:12) is *rnsoc:renarratedBy* (Line:27) *c:me* (Line:27) that indicates the address of the *rnsoc:Renarrator* Bob's profile (Line:9).

5.1.7. Document

Document is a Web document that is subject to renarration. Renarration transformations in a renarration create atomic changes on the segments of this document by making use of DOM (Document Object Model). After the appliance of renarration transformations, the source document becomes a renarrated document, and these changes occur on the client-side through DOM usage. In the Figure 5.15 document, renarration, and narration classes and their relations can be seen.

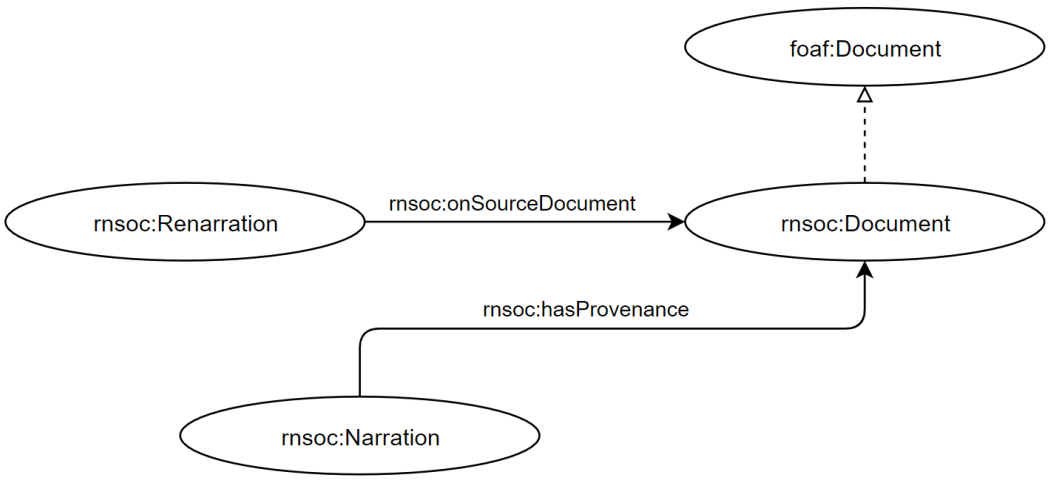

Figure 5.15. Document class and its significant relations

rnsoc:Document class is the subclass of *foaf:Document*. *rnsoc:Document* class has *rnsoc:onSourceDocument* relation with *rnsoc:Renarration* class. Therefore, *rnsoc:onSourceDocument* object property denotes the web resource that renarration belongs to, and this relation captures the origin. *rnsoc:hasProvenance* object property is used for referencing the external resource of *rnsoc:Narration* class. The Table 5.8 shows the document class and associated prominent elements.

Table 5.8. Description of document class and directly associated elements.

Term	Type	Description
foaf:Document	Class	Specifies a general document class [43]
rnsoc:Document	Class	rn:Document is a subClass of foaf:Document. It is a web resource that represents a digital document.
rnsoc:onSourceDocument	Object Property	Relates a renarration to a document that is subject to the renarration
rnsoc:hasProvenance	Object Property	The relation between a narration and external resource from which the content may be referenced.

Example Case: Jane navigates to a web document and sees a writing error in a bold text. She renarrates the document to remove the bold text to correct the mistake.

```
@prefix : <#>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix owl: <http://www.w3.org/2002/07/owl#>.
@prefix rnsoc: <http://soslab.cmpe.boun.edu.tr/ontologies/rnsoc.owl>.
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.
@prefix oa: <http://www.w3.org/ns/oa#>.
@prefix dc: <http://purl.org/dc/elements/1.1/>.
@prefix ren: <https://neua.gitlab.io/renarration-test-cases/>.
@prefix c: <https://jane-therenarrator.example.com/profile/card#>.

:kfb3de1a-2ce5fad3-f1e1b21a
  a owl:NamedIndividual, rnsoc:Renarration;
  rnsoc:createdAt "2020-11-06T13:54:11.002Z"^^xsd:dateTime;
  rnsoc:hasMotivation "correction"^^xsd:string;
  rnsoc:hasRenarrationTransformation
    [
      a rnsoc:RenarrationTransformation;
      oa:hasSelector
        [
          a oa:XPathSelector;
          rdf:value "/html[1]/body[1]/div[3]/p[1]/b[1]"
        ];
      rnsoc:createdAt "2020-11-06T13:51:11.002Z"^^xsd:dateTime;
      rnsoc:hasAction "remove"^^xsd:string;
    ];
  rnsoc:onSourceDocument ren:document_example;
  rnsoc:renarratedBy c:me.
ren:document_example a rnsoc:Document.
```

Figure 5.16. A renarration specification that makes a correcting by removing an element

Figure 5.16 shows the renarration specification for the document example case. The *rnsoc:Renarration* (Line:12) has the *rnsoc:onSourceDocument* (Line:26) of *ren:document_example* (Line:26) that points out the address of the *rnsoc:Document* (Line:8). The Line:28 indicates that the *ren:document_example* is a *rnsoc:Document*.

6. PROTOTYPE IMPLEMENTATION

The prototype is a proof of concept social renarration platform that demonstrates the usability of the proposed Renarration Social Ontology that represents the renarrations and social interactions. All users are considered to be potential renarrators, therefore all users are referred to as renarrators regardless of whether they have created a renarration or not.

6.1. Technologies

The prototype RNEx [16] is a browser extension that utilizes HTML [35] markup language, CSS [36] style sheet language, and JavaScript [37] programming language. RNEx utilizes Solid Project [19] to store and process renarrations, activities, and responses while conforming to linked-data [70] principles, and actualizing the decentralization and privacy protection aspect. RNEx uses rdflib.js [38] to read and write renarration, activity, response data expressed in Turtle [31] file format and to send and receive these files from Solid Pods (personal online data stores). RNEx utilizes Renarration Social Ontology to create and read the structured renarrations, activities, and responses data.

Users, namely renarrators, need to authenticate their Solid Pods with their WebIDs to sign in to the RNEx. Libraries solid-auth-client [40] and solid-file-client [71] facilitate to benefit from Solid authentication mechanism in the RNEx. The authorization mechanism of Solid utilizes WAC (Web Access Control) [20], a decentralized cross-domain access control system, that enables renarrators to control the access of others to their renarrations and responses, thus they may limit the people who have the right to view their renarrations and responses.

6.2. System Architecture

The system architecture contains four main subcomponents: central Solid pod, Solid pods of renarrators, Renarration Social Ontology, and RNEx. The prototype RNEx utilizes the Solid project [63] for the server-side operations, namely as a server and database for storing semantic information in the Turtle form [32]. The client-side of RNEx handles DOM manipulations, semantic information creation using Renarration Social Ontology, and communication with the appropriate Solid pods, i.e. central solid pod and solid pods of renarrators.

RNEx-Central Solid pod stores the creation information, index, of renarrations in a turtle file named RenarrationList. In other words, inside the RenarrationList turtle file triples indicating the WebID of renarrator, URL of renarration, URL of document, and creation time information is stored. To preserve privacy the RenarrationList file doesn't contain the renarration content e.g., renarration transformations, selectors, actions, languages, motivations.

Similarly, RNEx-Central stores the creation information, index, of responses i.e. comments, rates in a turtle file named ResponseList. ResponseList turtle file contains triples denoting the WebID of renarrator, URL of response, URL of renarration, and creation time information. To preserve privacy ResponseList file doesn't contain the response content, e.g., comment value, rate value. The Figure 6.1 shows the system architecture for renarration creation. It contains the communication of the RNEx browser extension with Solid pods of the RNEx-Central and renarrators, and usage of the Renarration Social Ontology, and creation of a renarration for a source document.

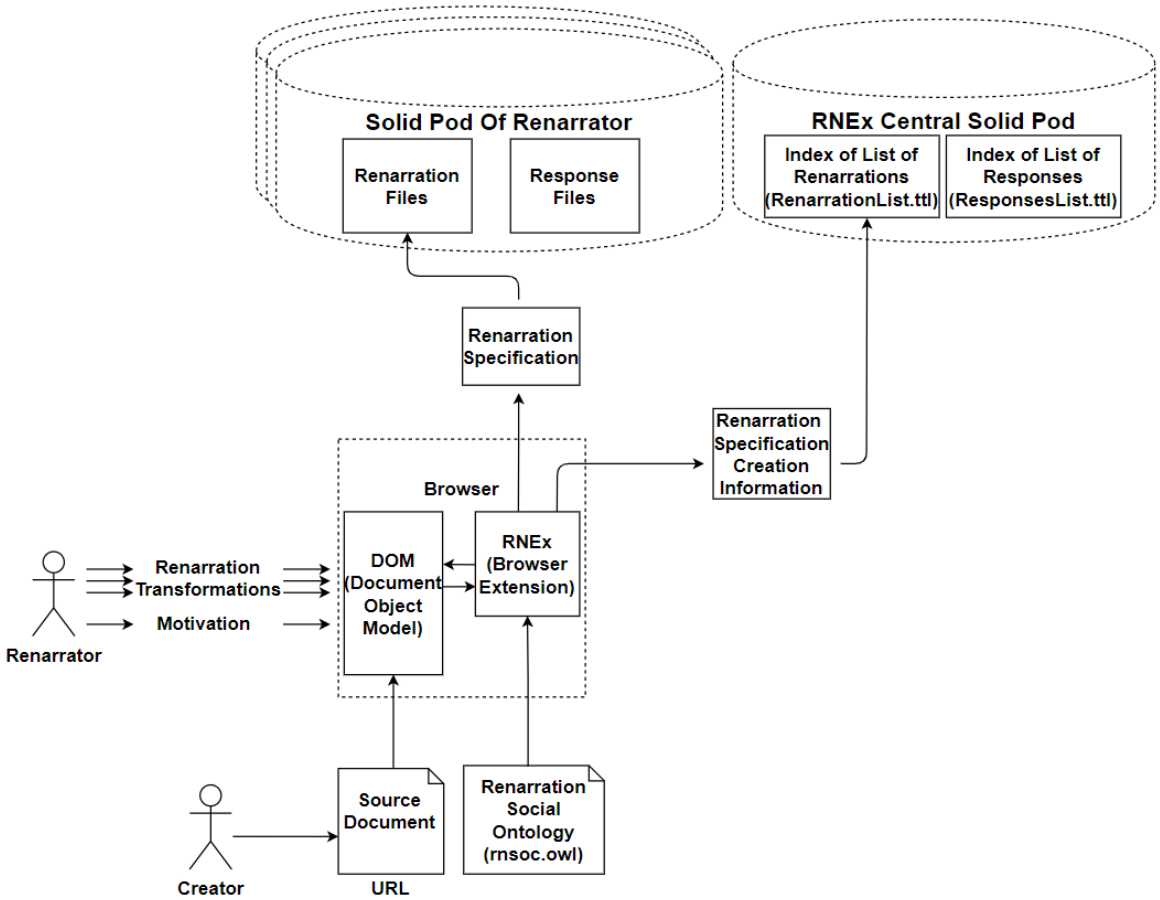

Figure 6.1. System architecture for renarration creation

When a renarrator wants to create a renarration of a source document, they should sign to RNEx with their Solid accounts to enable communication of RNEx with their Solid Pods. Then, they create a renarration on the desired source document by restructuring the DOM. The changes on the DOM are stored in a renarration file conforming to Renarration Social Ontology. Then, RNEx sends the renarration file to Solid pod of the renarrator, and triples indicating the creation of this renarration to RenarrationList file on the RNEx-Central pod.

When a renarrator chooses to view a renarration of a source document among others, RNEx retrieves URIs of the list of renarrations about the source document from RenarrationList on the RNEx-Central pod. Then, RNEx retrieves these renarrations from the Solid pods of corresponding renarrators. Then, the renarrator chooses a renarration to view, and RNEx applies it to DOM.

6.3. Design Of Solid Pods

RNEx-Central Solid pod stores the creation information, index, of the renarrations and responses. Method of holding the creation data of renarrations and responses in a central pod simplifies finding renarrations about a web document, renarrations of a renarrator, and responses about a renarration. This method complies with the privacy aspect of RNEx, because the content of renarrations and responses are stored in the Solid pod of renarrators, in other words, the substances of renarrations and responses are not stored in the RNEx-Central. Respectively the creation information, index, of renarrations are stored in the RenarrationList.ttl, and responses are stored in the ResponseList.ttl. In the Figure 6.2 creation information of renarrations inside the RenarrationList.ttl turtle file can be seen, similarly Figure 6.3 shows the ResponseList.ttl.

The central pod method facilitates finding renarrations, because not using a central pod, consecutively getting the renarration information from Solid pods of connected renarrators, causes various problems, e.g. hard to find disconnected renarrations, long-lasting retrieval of renarrations, and escalating complexity due to search of all the renarration network during retrieval operations.

To implement above mentioned central pod method and separate the functionality of the RNEx-Central Solid pod and renarrators' Solid pods, creation information of the renarration and responses are stored in the RNEx-Central pod, and content of renarrations and responses are stored in the Solid pods of renarrators. Container and resource structure of the RNEx-Central and renarrators' pods can be respectively seen in the Figure 6.4 and in the Figure 6.5.

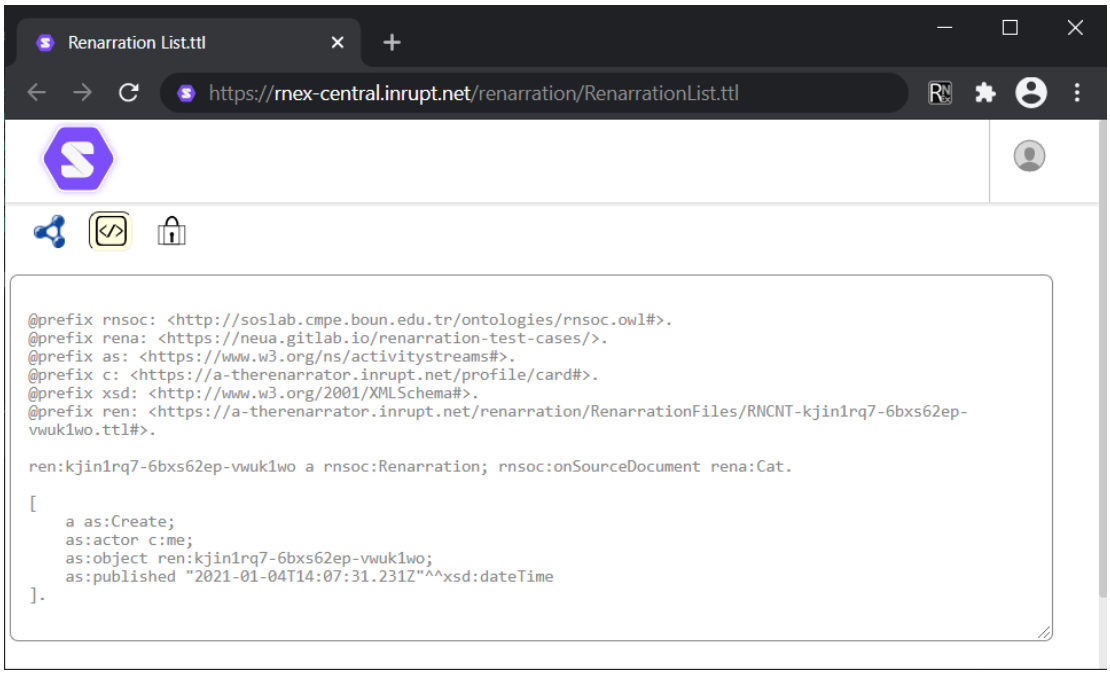

Figure 6.2. Creation information of renarrations inside the example RenarrationList

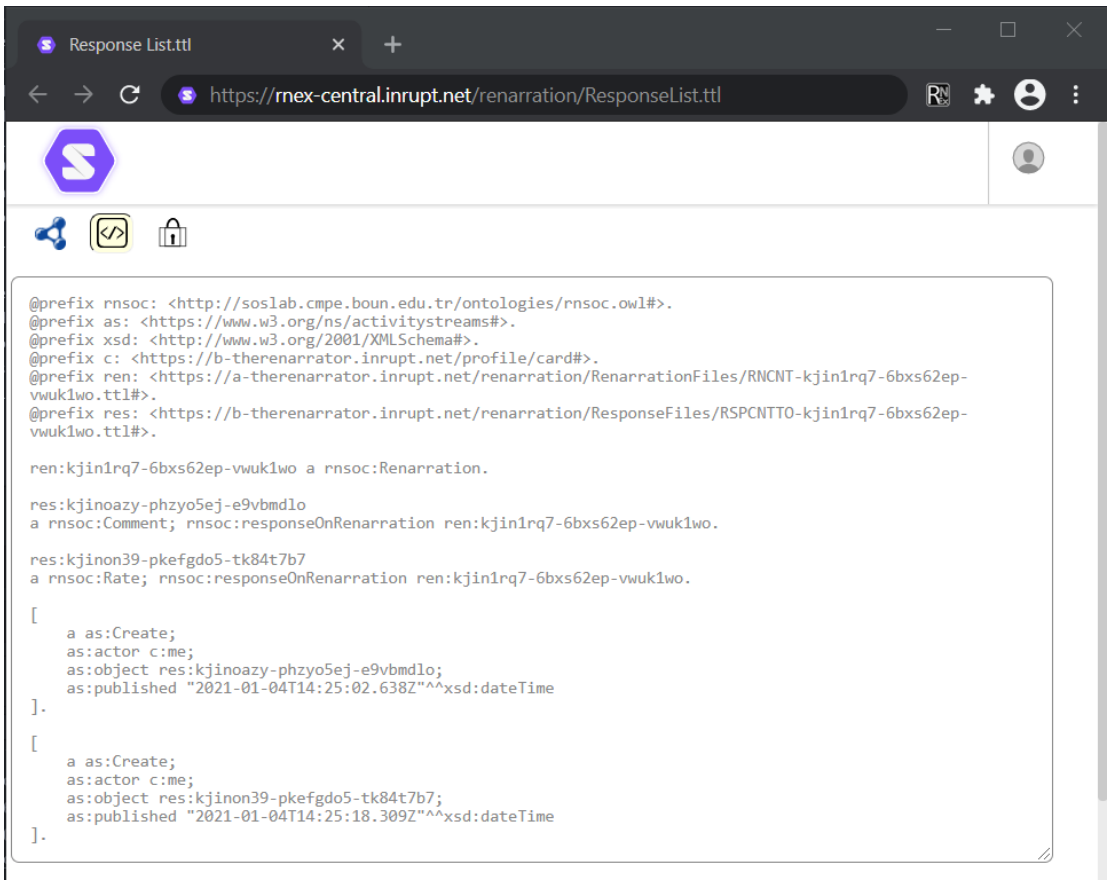

Figure 6.3. Creation information of responses inside the example ResponseList

In the Figure 6.2 an example data of a renarration stored in the RenarrationList can be seen. Normally, RenarrationList resource contains many of such triples depending on the number of total renarrration created on a social renarration platform. In the Figure 6.3 example of rate and comment responses data stored in the ResponseList can be seen. Normally, ResponseList resource contains many of such triples depending on the number of total responses created on a social renarration platform.

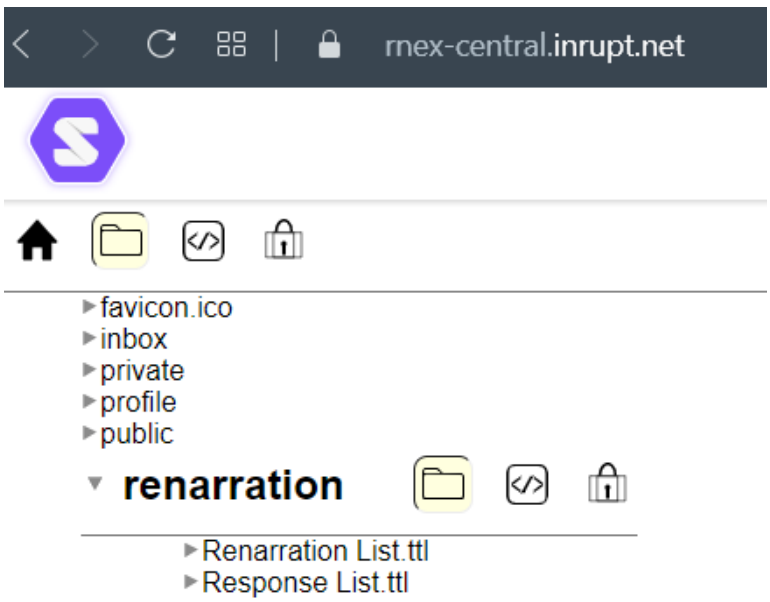

Figure 6.4. Renarration related containers and resources in the central pod

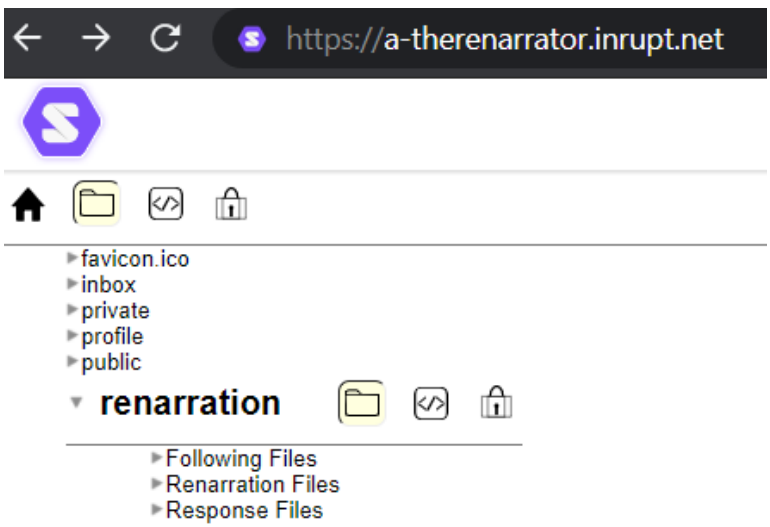

Figure 6.5. Renarration related containers and resources in a pod of renarrator

In the Figure 6.4 RNEx-Central Solid pod folders (containers) and files (resources) can be seen. The renarration folder and RenarrationList and ResponseList files inside it, indicates the path and URI of the renarration related resources. In the Figure 6.5 a renarrator's Solid pod folders (containers) and files (resources) can be seen. The renarration folder and FollowingFiles, RenarrationFiles, ResponseFiles inside it, indicates the path and URI of the renarration related resources. Inside the RenarrationFiles, the ResponseFiles, the FollowingFiles folders; renarrations of the renarrator, responses of the renarrator, other renarrators followed by the renarrator are stored, respectively.

6.4. Authentication and Authorization

RNEx obtains authorization to read and write semantic data from the Solid pod of a renarrator when the renarrator signs to their Solid pod using the unique WebID. In the Figure 6.6 authentication to Solid pod with the prototype can be seen.

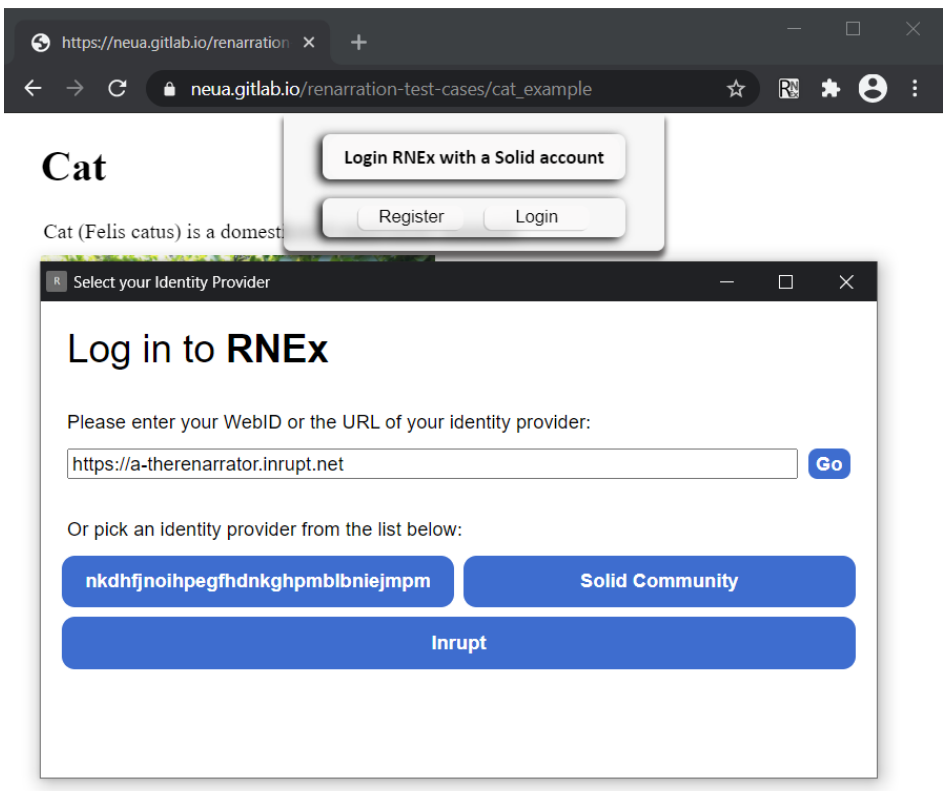

Figure 6.6. Authentication to Solid pod with the prototype

A renarrator can access the information in the pod of others, i.e. renarrations and responses, depending on the ACLs [20] (Access Control List) of these resources. ACLs give renarrators the possibility to restrict sharing their containers and resources in their Solid pods e.g., renarrations, responses. The ACL usage of Solid pods empowers the privacy aspect of the RNEx extension.

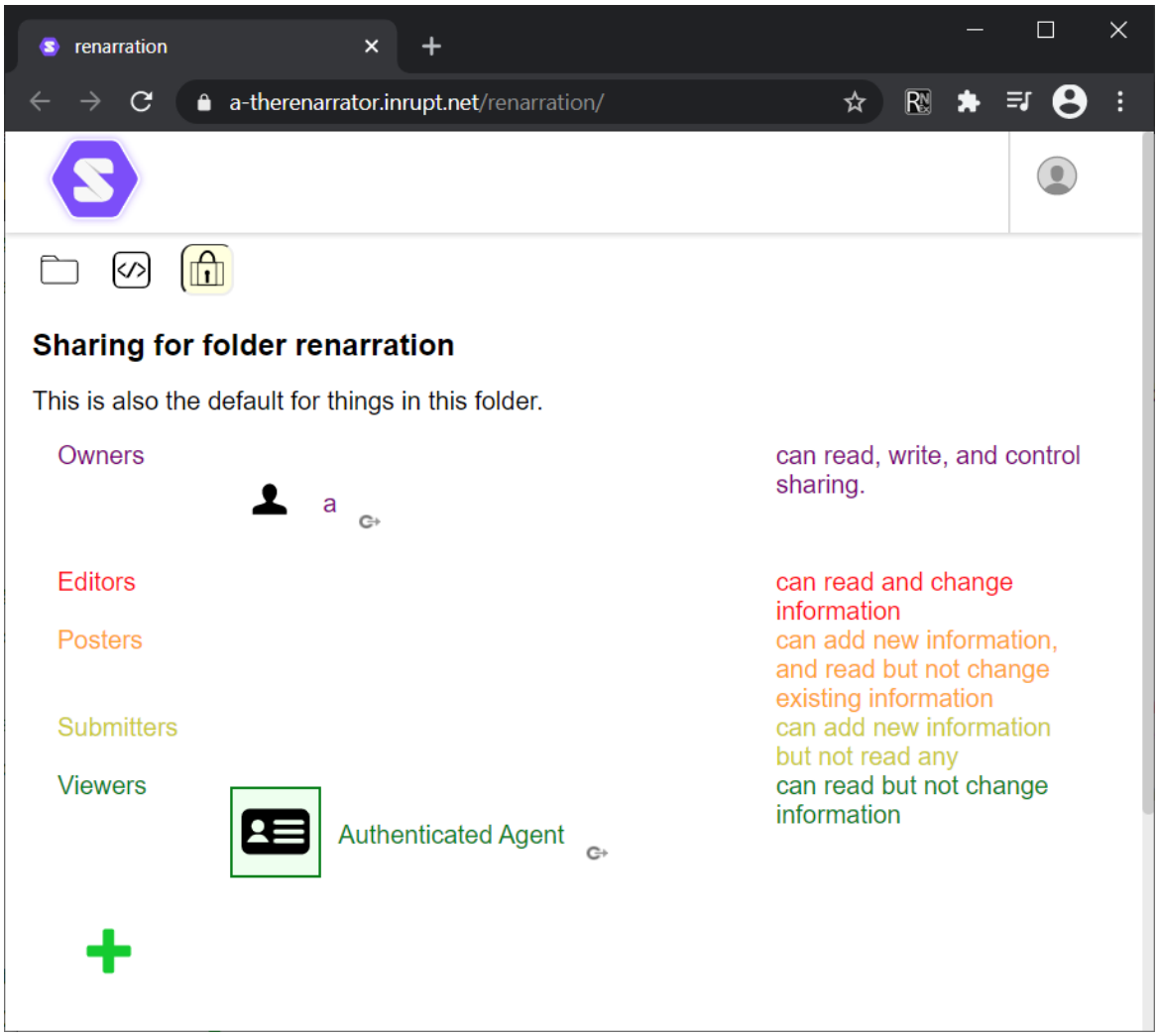

Figure 6.7. Authorization to the renarration container in a Solid pod

Renarrators can authorize others to read, write and control their renarrations with the use of Solid. In the Figure 6.7 authorization to the renarration container in a pod can be seen, "a-therenarrator" lets other authenticated agents to view the renarration folder, and the ACL file of it can be observed in the Figure 6.8.

```
@prefix : <#>.
@prefix acl: <http://www.w3.org/ns/auth/acl#>.
@prefix ren: <./>.
@prefix c: </profile/card#>.

:ControlReadWrite
    a acl:Authorization;
    acl:accessTo ren:;
    acl:agent c:me, <mailto:example@example.com>;
    acl:default ren:;
    acl:mode acl:Control, acl:Read, acl:Write.
:Read
    a acl:Authorization;
    acl:accessTo ren:;
    acl:agentClass acl:AuthenticatedAgent;
    acl:default ren:;
    acl:mode acl:Read.
```

Figure 6.8. ACL file of renarration container

6.5. Renarration Implementation

6.5.1. Renarration Creation

In the foreground, to create a renarration, renarrator types an URL into the address bar of the browser and navigates to a chosen web page, and launches the RNEx browser extension then chooses the icon for creating renarration in the UI (User Interface). Renarrator selects a fragment of the web page and decides what to do with this selected fragment by choosing an action i.e., "remove", "insert", "replace", and depending on the chosen action renarrator types a content. If the "replace" action is chosen, the renarrator creates new content for replacement with the selected fragment. If the "insert" action is chosen, the renarrator creates content to insert before or after the selected fragment. If the "remove" action is chosen, the selected fragment is removed. Lastly, the renarrator types a language code before finishing the renarration transformation. Afterwards, renarrator presses the save the renarration transformation button to finish the renarration transformation.

Renarrator can create many renarration transformations by repeating the above-defined process, before finishing the renarration, renarrator chooses a motivation then presses the save the renarration button to conclude it. In the Figure 6.9 renarration creation with the prototype can be seen. Numbers near the dotted red boxes in the Figure 6.9 depict the UI of RNEx. Number 1 is the button for activating and deactivating the extension. Number 2 is horizontal and vertical menus and toggle button to active different functionalities. Number 3 is action selection part. Number 4 is HTML of chosen element. Number 5 is toggle button for activating writing the HTML as an input. Number 6 is area for the content of narration that lets renarrator to type text, or add image, audio or video. Number 7 is language of the renarration transformation. Number 8 is button for saving the renarration transformation. Number 9 is part for choosing motivation for the renarration. Number 10 is button for finishing and saving the renarration. Number 11 i selected fragment in the HTML document, i.e, object of a selector.

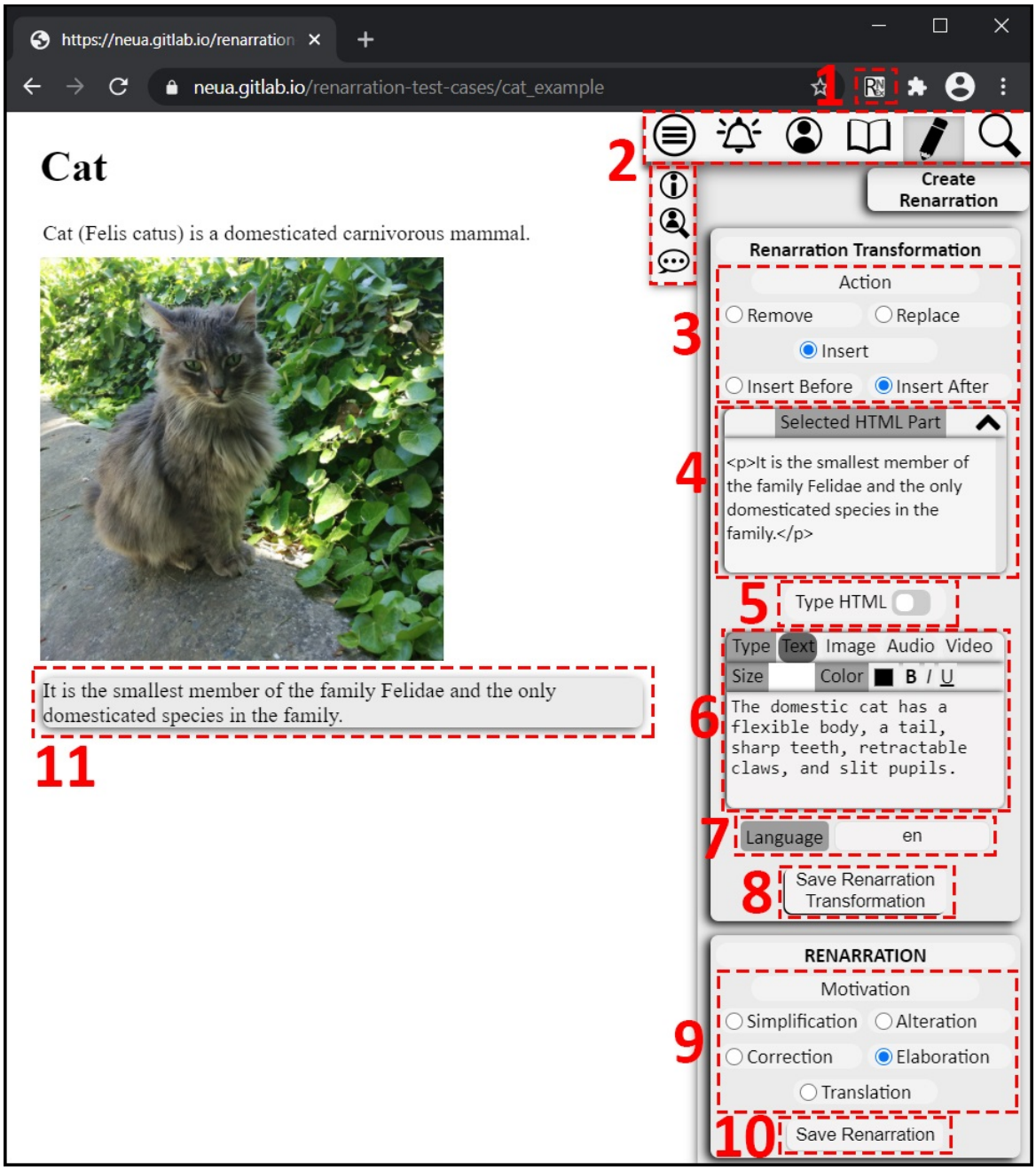

Figure 6.9. Renarration creation with the prototype

The sequence diagram (Figure 6.10) shows the background processes for renarration creation. When the renarrator chooses the icon for creating renarration, RNEx gets the URL of the chosen web page and WebID of the renarrator (startRenarration). RNEx gets the action, selection, and narration from the renarrator each time renarrator creates a renarration transformation (getAction, getSelection, getNarration).

RNEx depending on the chosen action i.e., "replace", "insert", "remove" respectively either replaces the selection (selected HTML fragment) with the narration (replaceDomSelection), or inserts a narration to the selection (insertDomSelection), or removes the selection (removeDomSelection).

RNEx currently supports XPathSelector selections and it doesn't support TextPositionSelector selections. RNEx repeats these steps until the renarrator finishes all the renarration transformations. Renarrator selects a motivation before finishing the renarration (setMotivation). Then, RNEx sets the creation time of the renarration (setTimeOfRenarration).

RNEx uses the rdflib.js to create the turtle triples of corresponding operations. It creates a turtle file of the completed renarration with a unique id on the renarrator's Solid pod (saveRenarration). Then, it updates the RenarrationList.ttl file on the RNEx-Central Solid pod with triples indicating the URI of the renarration, renarrator WebID, URI of the source document, and creation time (saveRenarrationInfo).

In the Figure 6.11 a renarration turtle file on the Solid pod of the renarrator can be seen. Inside the renarration file *rnsoc:createdAt* property indicates time of the creation of renarration, *rnsoc:hasMotivation* indicates the purpose of the renarration, *rnsoc: hasRenarrationTransformation* shows the various renarration transformations that are created.

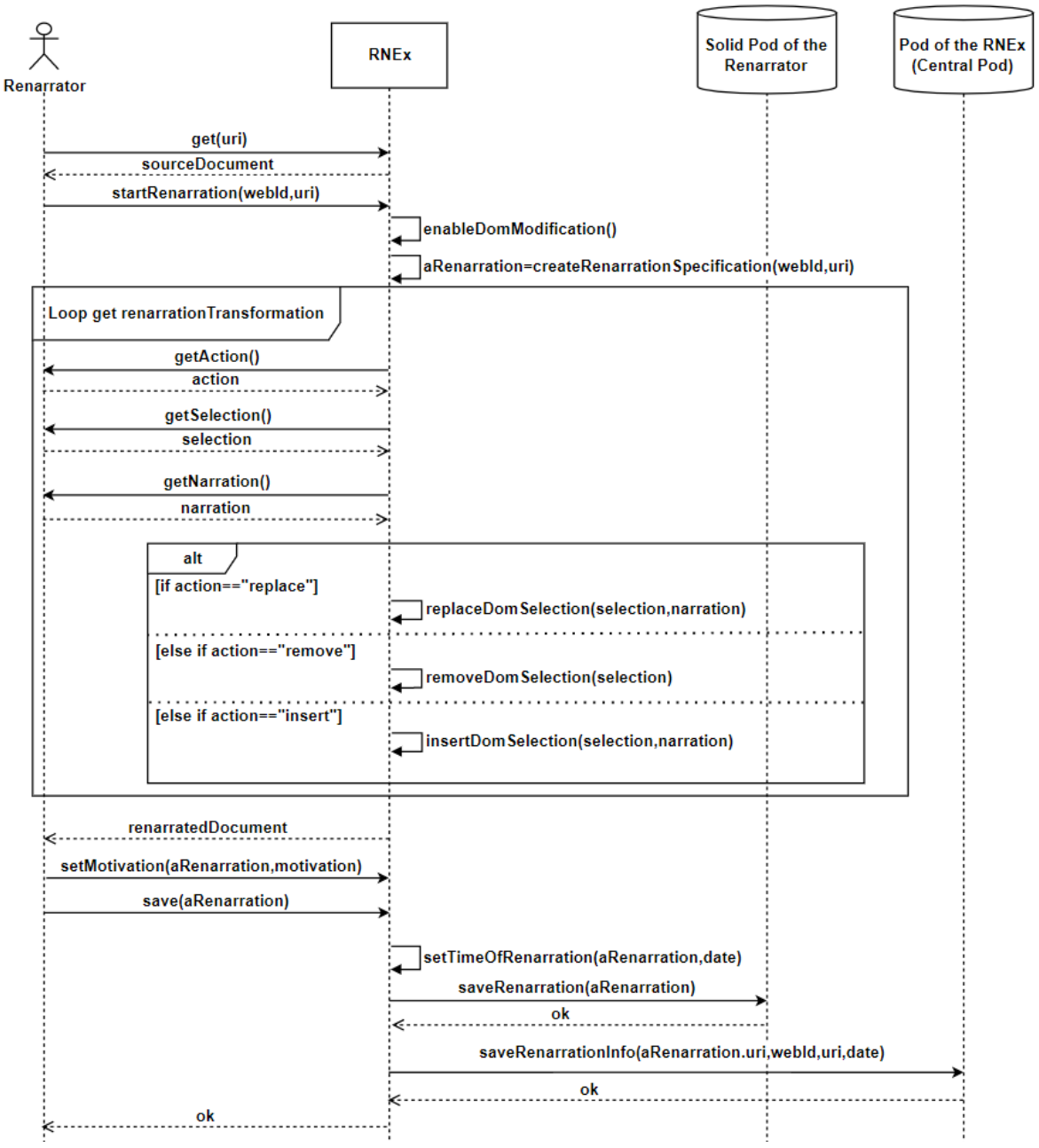

Figure 6.10. Sequence diagram for creation of a renarration

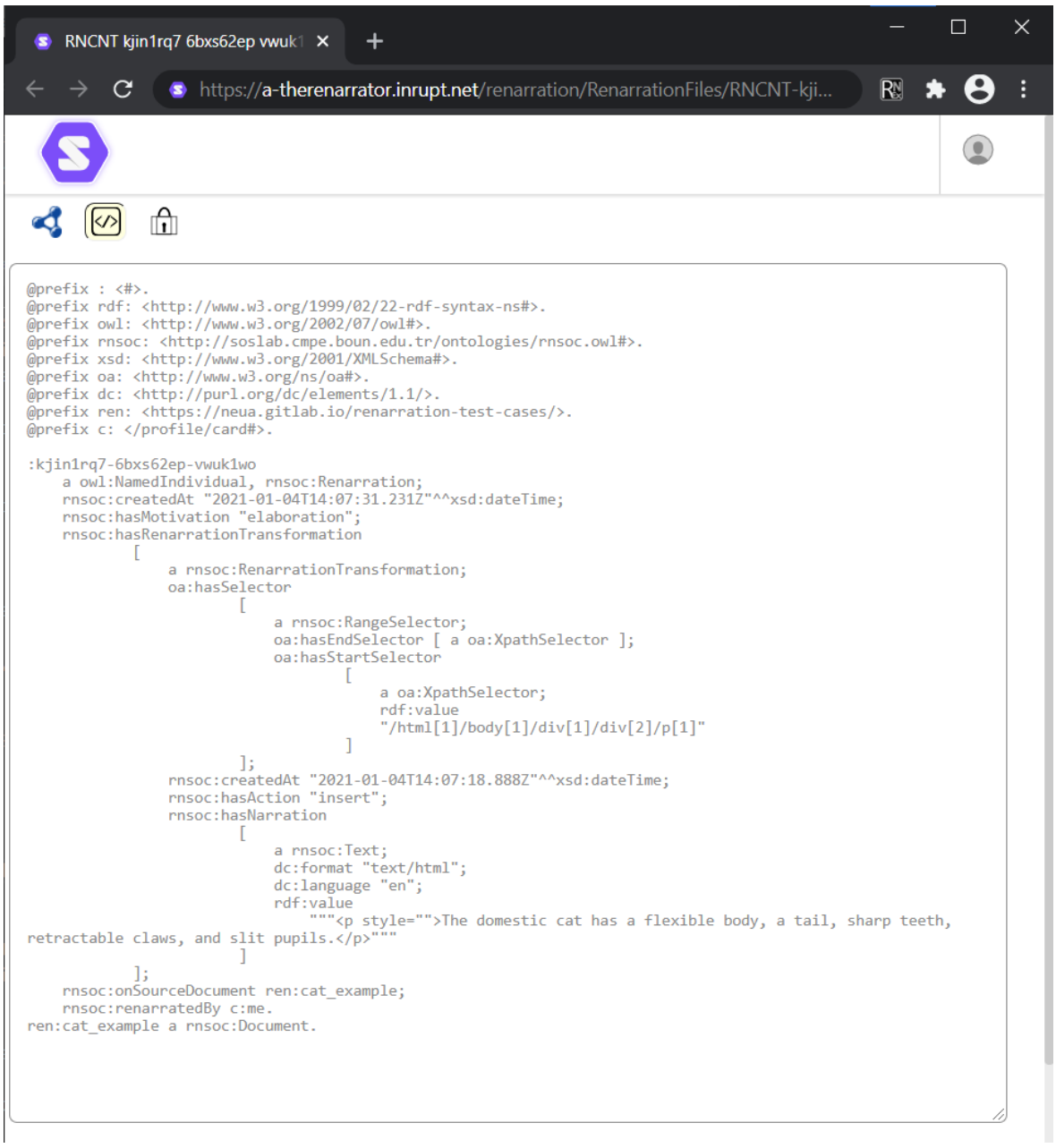

Figure 6.11. Renarration turtle file on the Solid pod of the renarrator

oa:hasSelector indicates the type of the selector for the renarration transformation, *oa:XpathSelector* points out the chosen fragment that is subject to a *rnsoc:hasAction*, *rnsoc:hasNarration* signifies the new content created by renarrator, *dc:language* indicates the language of the renarration transformation. *rnsoc:onSourceDocument* shows the original web document to apply the renarration and *rnsoc:renarratedBy* indicates the creator of the renarration.

In the Figure 6.12 renarration turtle triples on the central Solid pod can be seen. RenarrationList.ttl stores the triples related to each renarration creation. URI of the renarration indicates the location of the renarration, *rnsoc:onSourceDocument* shows the document that is subject the renarration, *as:Create* indicates the type of the activity, *as:actor* shows the actor of the create activity, *as:Object* indicates the URL of the created renarration, *as:published* shows the time of the create operation. All these triples informs about the location, creator, creation time, and source document of the renarration.

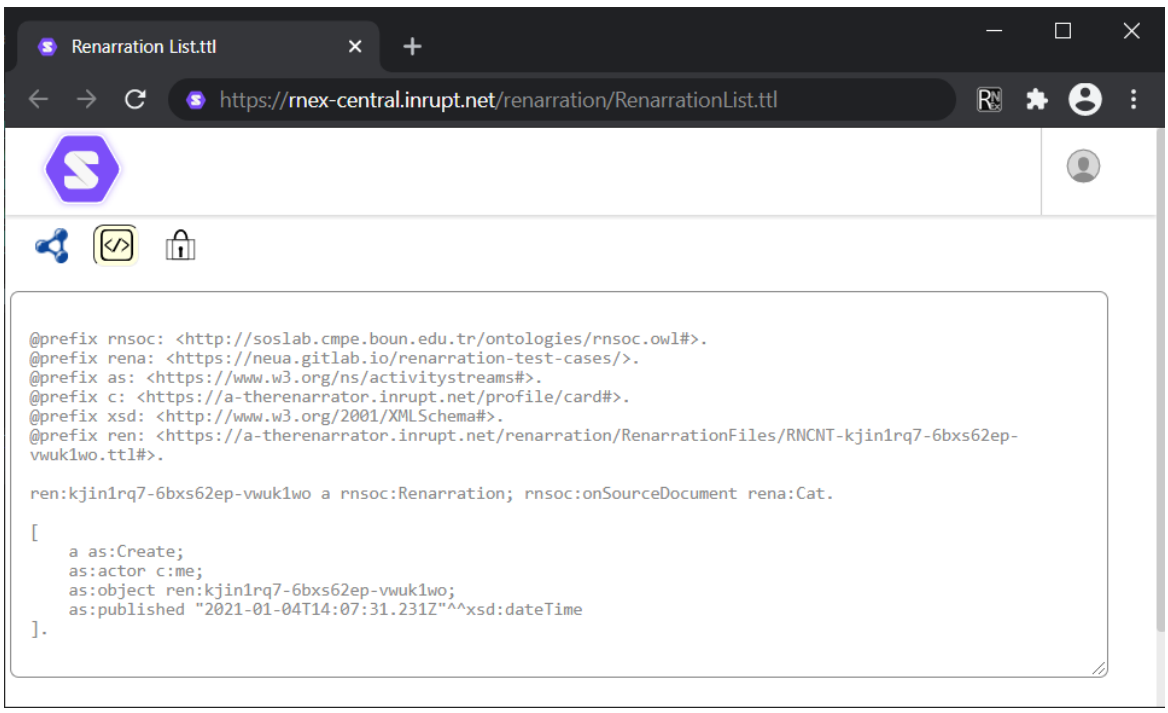

Figure 6.12. Renarration turtle triples on the central Solid pod

6.5.2. Renarration Retrieval

In the foreground, to view a renarration, renarrator types an URL into the address bar and navigates to a chosen web document, and after launching the RNEx extension, renarrator chooses the icon for reading renarration on UI. Renarrator can select a renarration among the renarrations that are created for the web document, by taking into consideration the motivation, language, renarrator, creation time of a renarration. RNEx then applies the preferred renarration to the web document, so that renarrator can view the renarrated document.

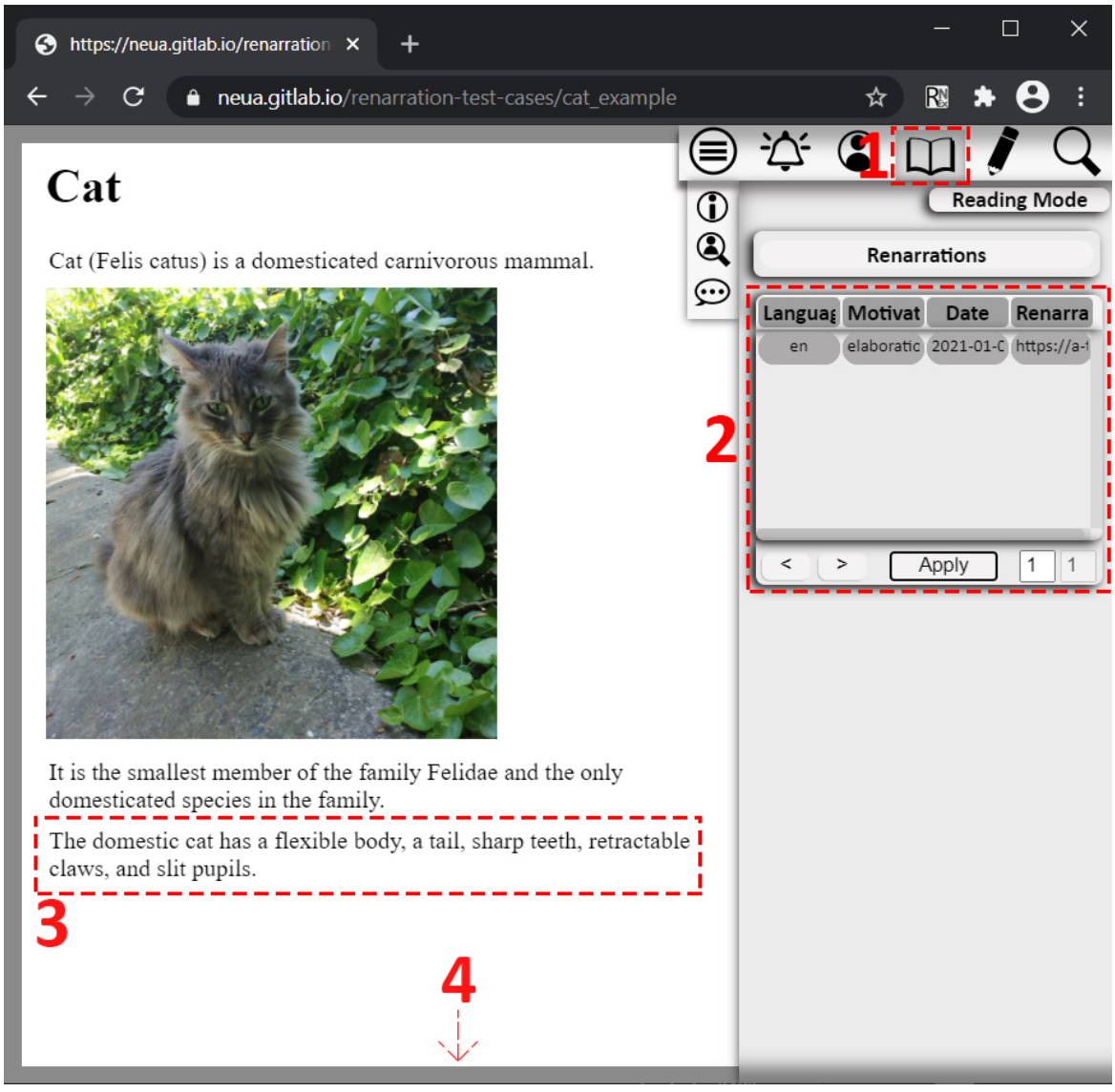

Figure 6.13. Renarration retrieval with the prototype

In the Figure 6.13 renarration retrieval with the prototype can be seen. Numbers near the dotted red boxes and the arrow in the Figure 6.13 depict the UI of RNEx. Number 1 is the button for activating the reading mode that has the functionality of showing the renarrations about the visited web document. Number 2 is renarrations table, each row corresponds to a renarration. Number 3 is the change in the source document after the renarration process. Number 4, the gray border around the screen, indicates that renarrated document is being viewed.

The sequence diagram (Figure 6.14) shows the background processes for viewing a renarration. When the renarrator chooses the icon for viewing a renarration, RNEx gets the URI of the preferred web page (source document). Then, RNEx searches triples related to source document inside the RenarrationList.ttl file which is on the RNEx-Central Solid pod (getRenarrationURIs). Then, RNEx finds triples about source document and URI of the related renarrations and their renarrators' WebID, and creation time (renarrationURIList). Then, RNEx retrieves the renarration specifications from the pods of the renarrators who renarrated the source document if the renarrator has the right to view it (getTheRenarration). RNEx stores these renarration specifications and their address to show them later (renarrations.add). This retrieval operation is handled with pagination, therefore RNEx gets number of renarration specifications from the pods. Each time renarrator request to see more renarrations it retrieves more renarrations related to the source document. The information obtained from these renarration specifications enables RNEx to show the list of renarrations about the source document to the renarrator (renarrations). Then, the renarrator chooses a renarration among the list of renarrations to view it (showRenarration). RNEx obtains the renarration transformations from the renarration specification (getRenarrationTransformations). Then, RNEx applies each renarration transformation that the renarration contains by taking the creation time of them into consideration. RNEx restructures the DOM in accordance with the selectors, actions and narrations of renarration transformations to generate the renarrated document (replaceDomSelection, removeDomSelection, insertDomSelection).

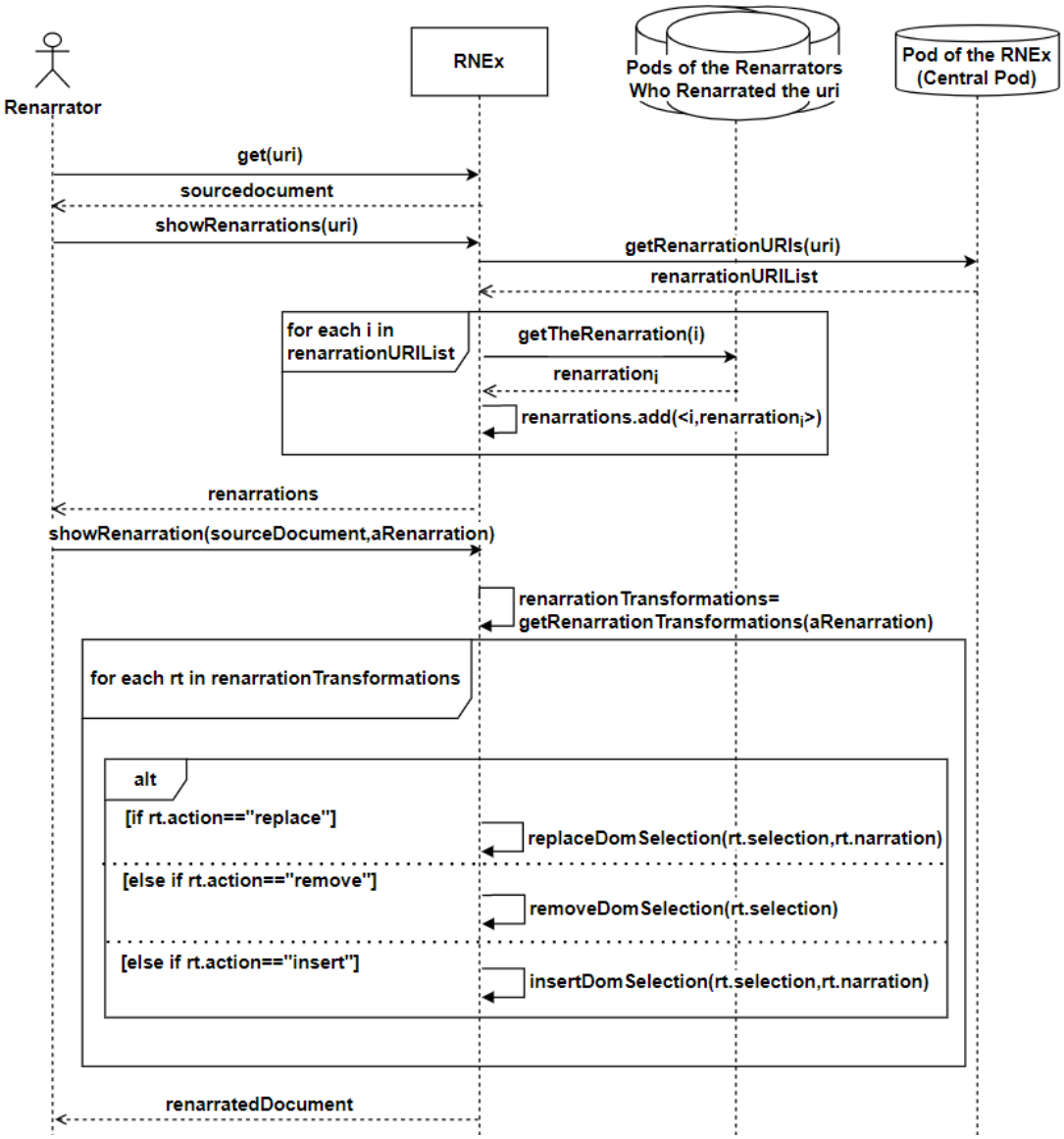

Figure 6.14. Sequence diagram for retrieval of a renarration

6.6. Response Implementation

6.6.1. Response Creation

In the foreground, to respond to a renarration, the renarrator chooses the icon for the response on UI of RNEx then either rates or comments the selected renarration. In the Figure 6.15 response creation with the prototype can be seen. Numbers near the dotted red boxes and the arrow in the Figure 6.15 depict the UI of RNEx. Number

1 is the button for activating the response mode that has the functionality of giving response on the renarration about the visited web document. Number 2 is the part for rating the renarration, each triangle corresponds to one point. Number 3 is the part for commenting the renarration. Number 4, the gray border around the screen, indicates that renarrated document is being viewed. Number 5 is the change in the source document after the applied renarration.

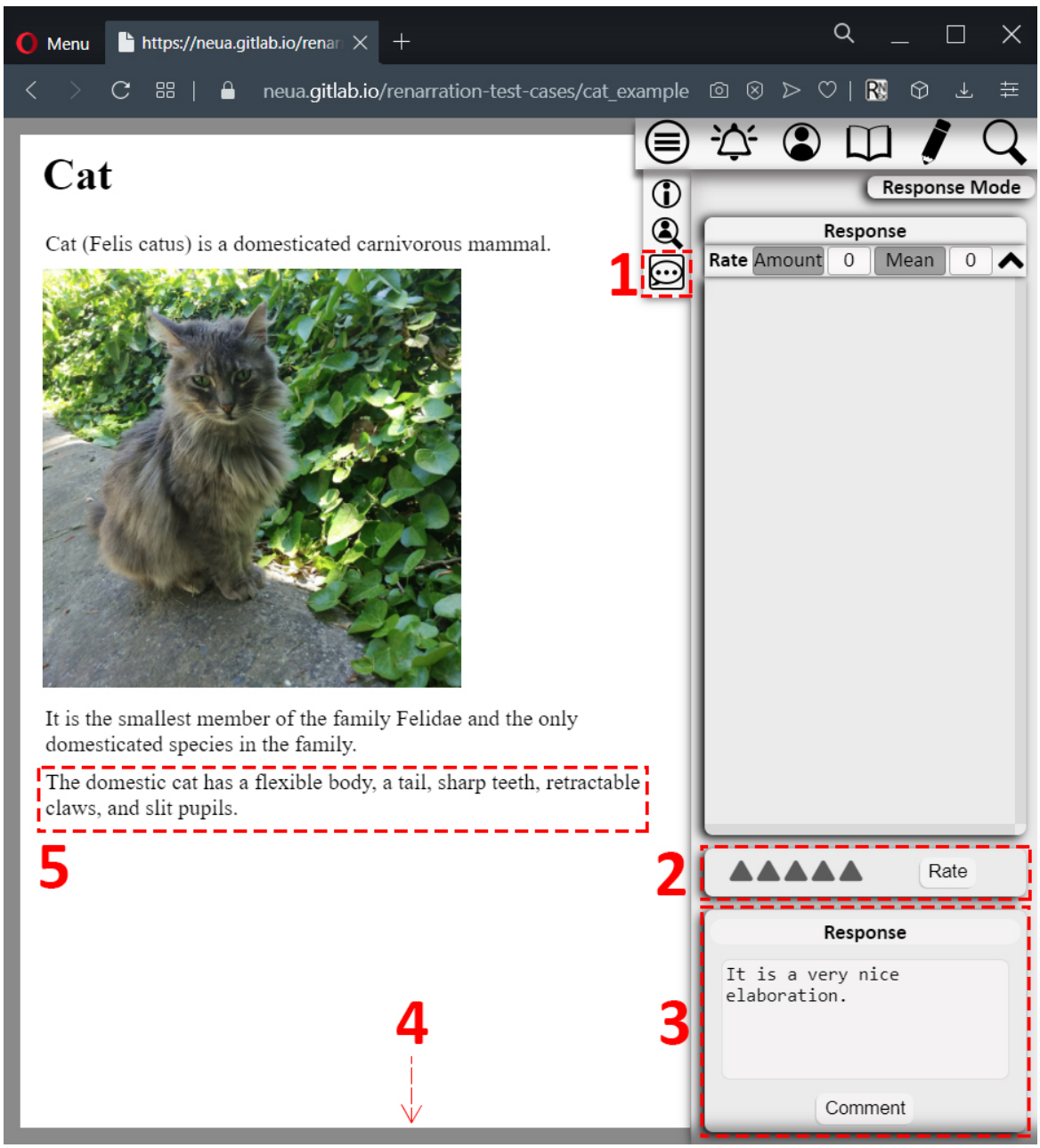

Figure 6.15. Response creation with the prototype

The sequence diagram (Figure 6.16) shows the background processes for creation of a response. Firstly, renarrator navigates to a source document and chooses a renarration to view (The part before the appearance of renarrated document in the Figure 6.16). When the renarrator creates a response on a renarration, RNEx gets the URI of the chosen renarration (createResponse). The response can either be a rate or a comment. Then, RNEx creates the corresponding triples according to the selected response, i.e., rate, comment using the rdflib.js and it generates a response URI (responseUri). Subsequently, the triples indicating the response are sent to the pod of the renarrator who created the response (saveResponse). Finally, the creation information of the response is sent to ResponseList.ttl file on RNEx-Central Solid pod (saveResponseInfo).

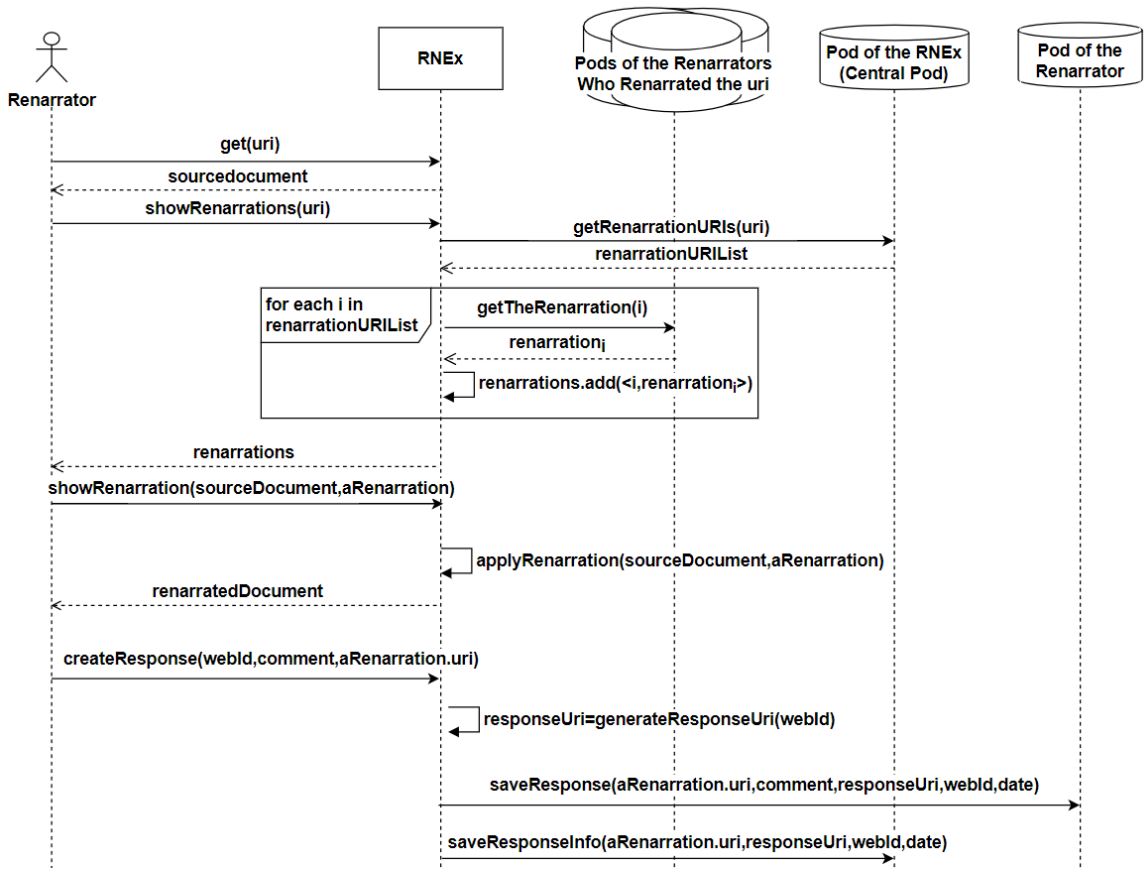

Figure 6.16. Sequence diagram for creation of a response

6.6.2. Response Retrieval

In the foreground, to retrieve responses of a renarration, the renarrator chooses the icon for the response on UI of RNEx then can see the rate and comments that are created for the selected renarration. In the Figure 6.17 retrieval of a response with the prototype can be seen.

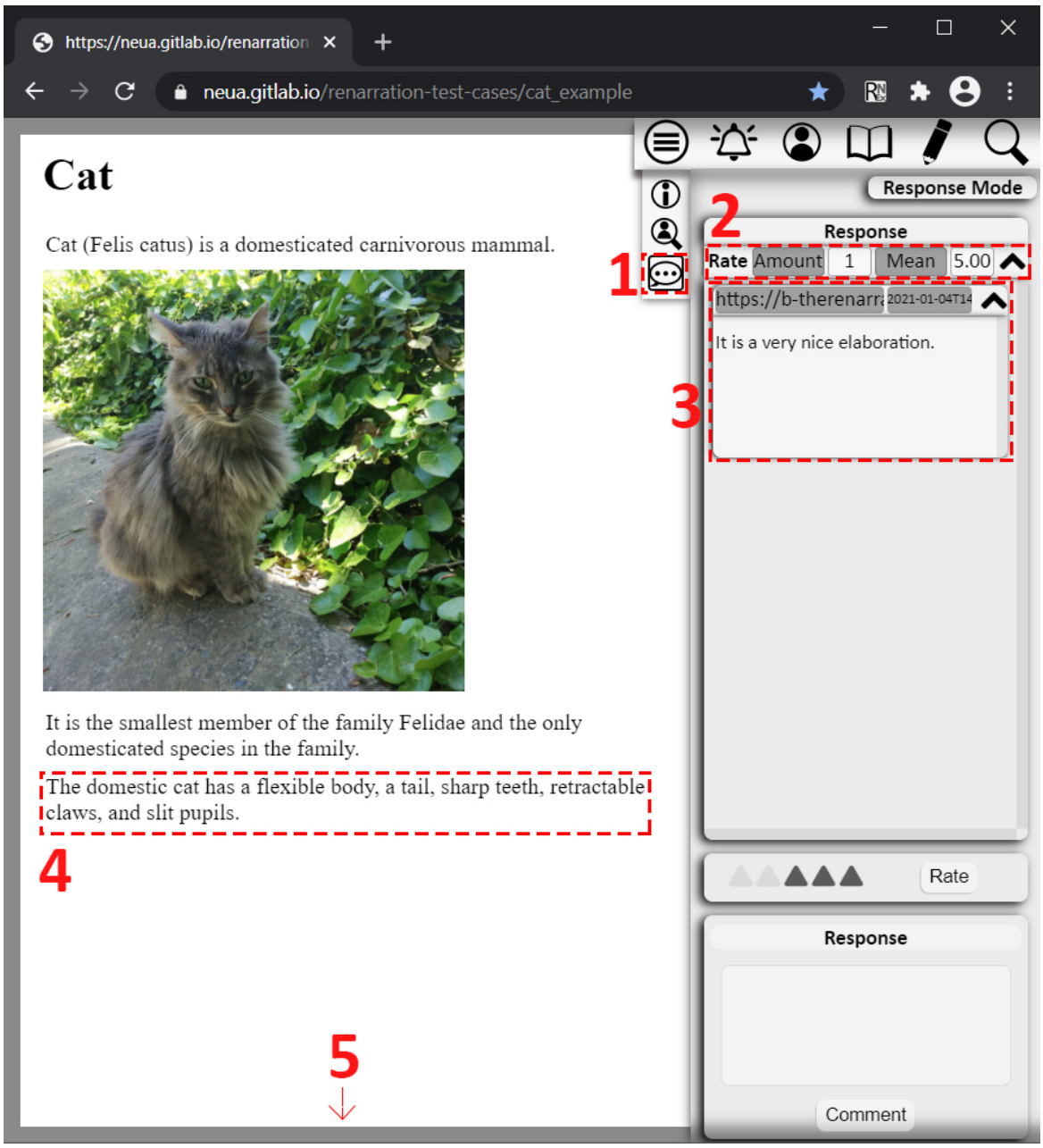

Figure 6.17. Response retrieval with the prototype

Numbers near the dotted red boxes and the arrow in the Figure 6.17 depict the UI of RNEx. Number 1 is the button for activating the response mode that has the functionality of giving a response on a renarration.

Number 2 is the part that shows the amount of rating and the mean for the viewed renarration. Number 3 is the part that shows the comment for the viewed renarration.

Number 4, the gray border around the screen, indicates that the renarrated document is being viewed. Number 5 is the change in the source document after the applied renarration.

The sequence diagram (Figure 6.18) shows the background processes for viewing responses. Firstly, renarrator navigates to a source document and chooses a renarration to view (The part before the appearance of renarrated document in the Figure 6.18).

When the renarrator requests to view responses about a selected renarration, RNEx gets the URI of the selected renarration (showResponses). Then, RNEx retrieves the matching information of responses from the ResponseList.ttl file of RNEx-Central Solid pod (getResponseURIs).

Solid pod locations of responses are derived by obtaining the matching triples of responses that are about the selected renarration (responseURIList). Then, RNEx retrieves the response specifications that the renarrator has the right to view, from the pods of the renarrators who responded to the renarration that is applied to the source document (getTheResponse). Lastly, the obtained responses are shown on the UI of RNEx (responses).

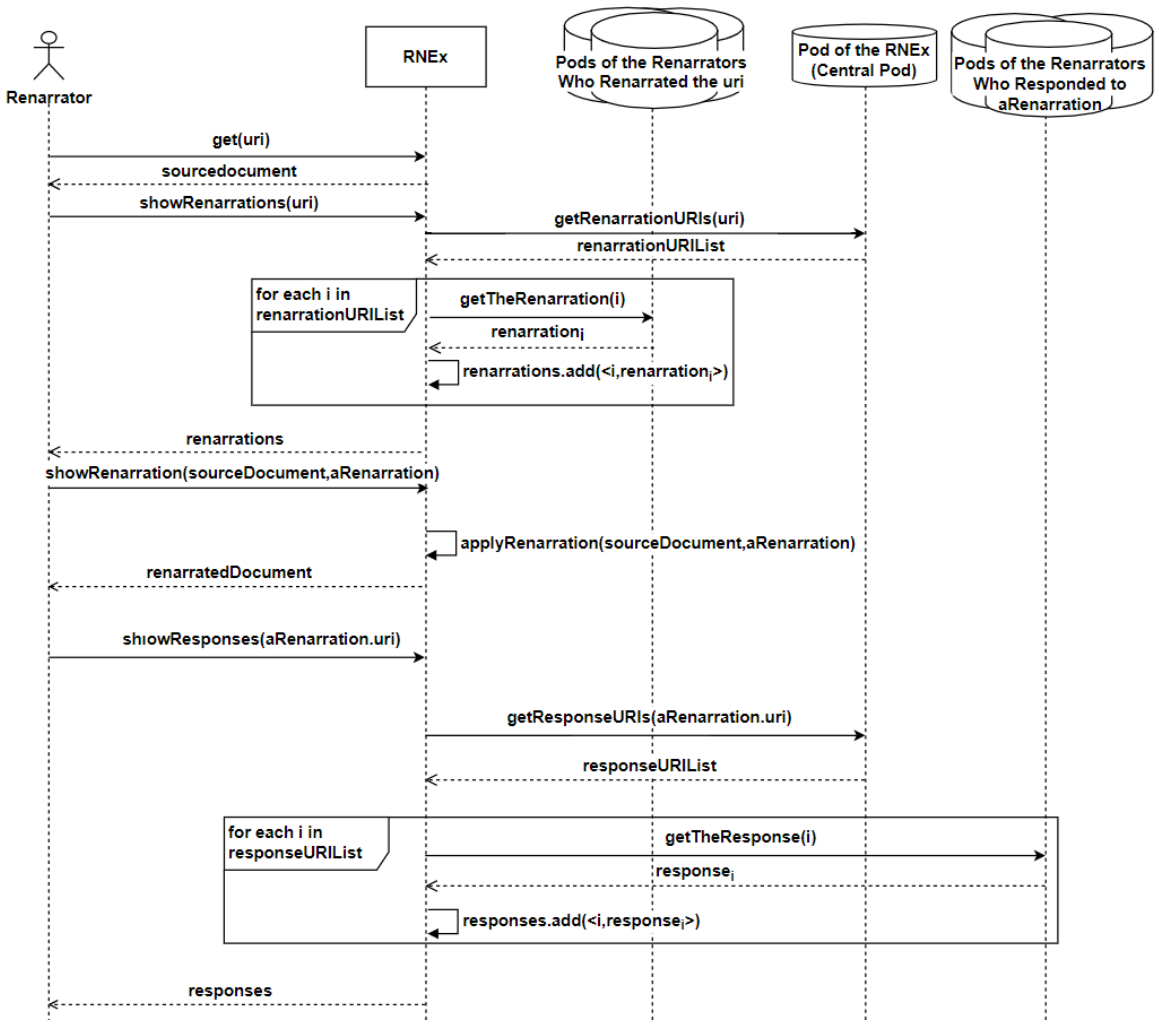

Figure 6.18. Sequence diagram for retrieval of responses

6.7. Following Implementation

To follow a renarrator, the logged in renarrator chooses the icon for the visited profile on RNEx extension UI, then can see the visited renarrator's renarrations and profile information and a button that is for following the visited renarrator. In the Figure 6.19 following a renarrator with the prototype can be seen.

Numbers near the dotted red boxes and the arrow in the Figure 6.19 depict the UI of RNEx. Number 1 is button for visiting the profile of the renarrator who created the current renarration that is applied to the web document. Number 2 is part that shows the profile information of the renarration e.g., name, WebID, role,

email. Number 3 is button for following the visited renarrator. Number 4 is part that shows the renarrations of the visited renarrator. Number 5 is the change in the source document after the renarration. Number 6, gray border around the screen, indicates that renarrated document is being viewed.

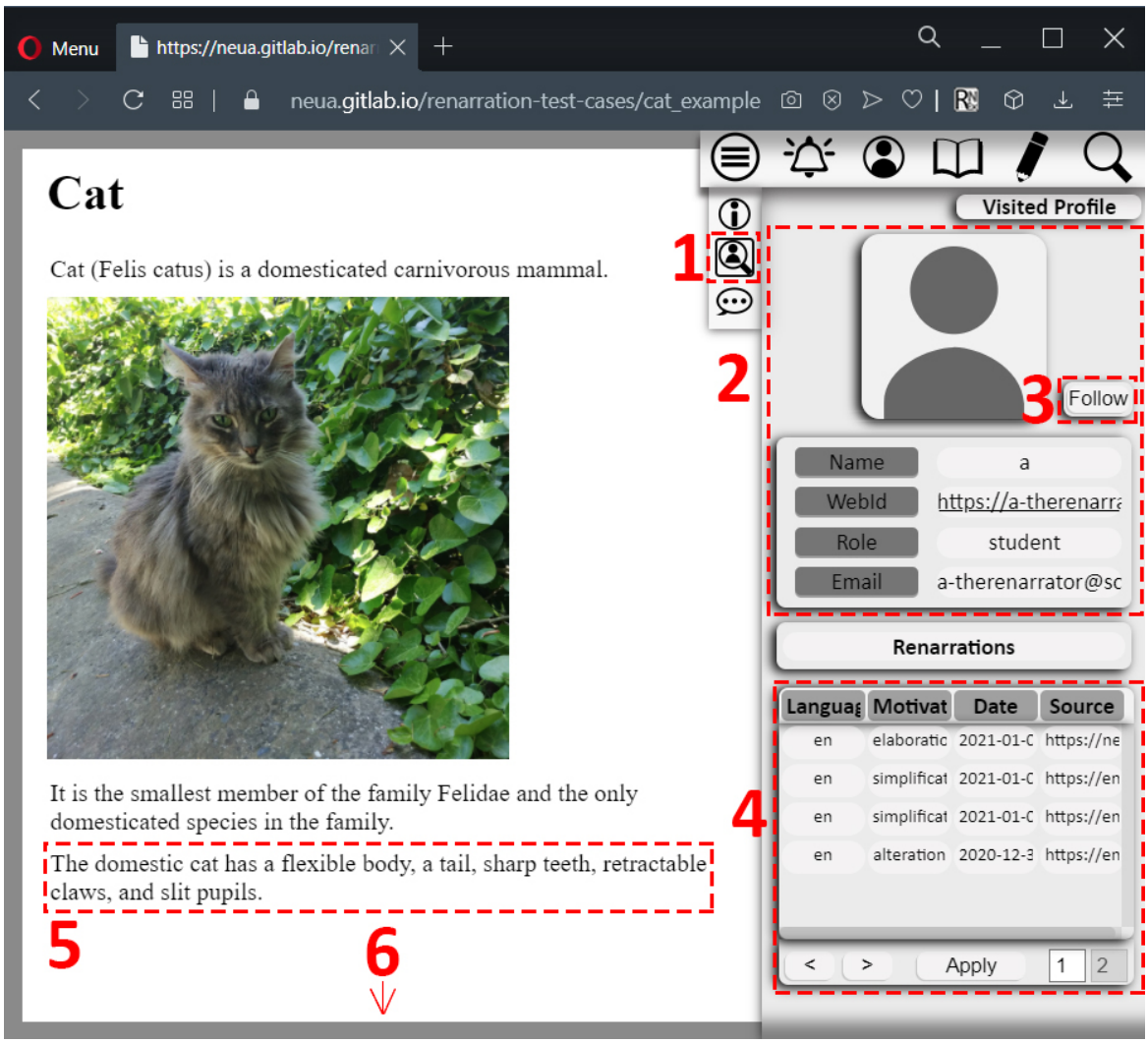

Figure 6.19. Following a renarrator with the prototype

In the background, when the renarrator follows other renarrator, RNEx gets the WebID of the renarrator who will be followed and creates triples representing this following process and adds to the triples on the Solid pod of the renarrator inside a FollowingList.ttl endpoint. In the Figure 6.20 FollowingList turtle file on a Solid pod can be seen.

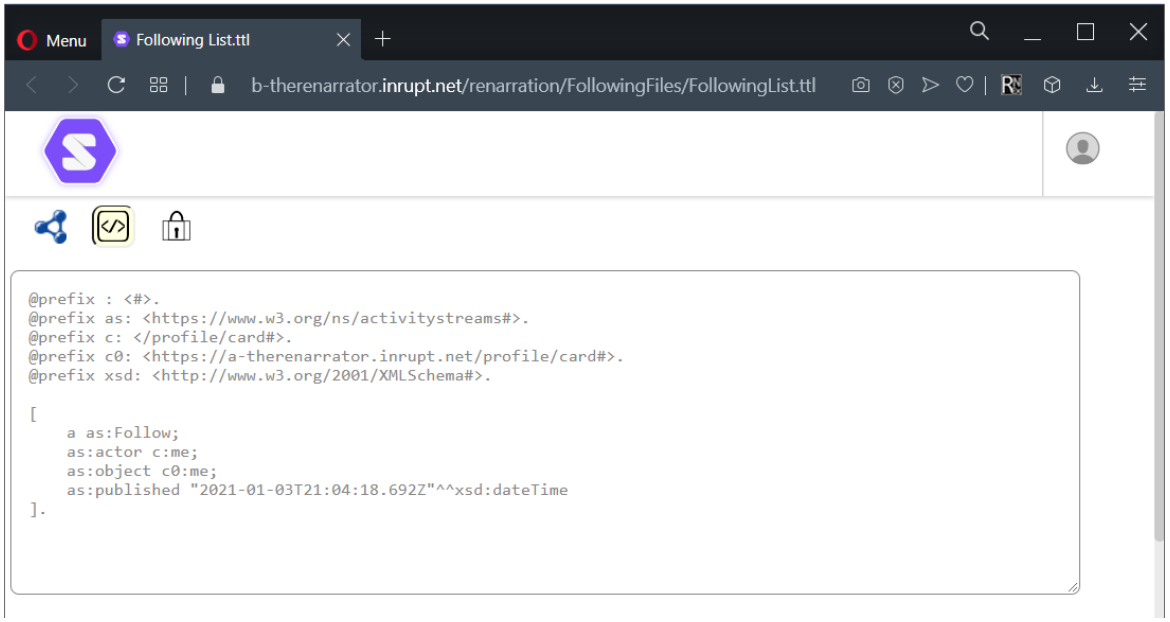

Figure 6.20. FollowingList turtle file on a Solid pod

6.8. Notification Implementation

To get notified about the followed renarrators, the renarrator chooses the icon for the notification on UI of RNEx then can see the notifications related to followed renarrators. In the Figure 6.21 notification functionality of the prototype can be seen. Current notification implementation is a basic way of notifying the renarrator.

Numbers near the dotted red boxes in the Figure 6.21 depict the UI of RNEx. Number 1 is button for viewing the notifications that shows the recent activities of followed renarrators. Number 2 is the box shows the source document URL, location, creation date of the renarration and WebID of its renarrator.

In the background, when the renarrator would like to see the notifications, RNEx gets the list of followed renarrators information from the FollowingList.ttl file which is in the pod of the current renarrator. Then, RNEx retrieves the RenarrationList.ttl and ResponseList.ttl files from the RNEx-Central to obtain renarrations and responses of the followed renarrators. Lastly, RNEx shows the renarrations and responses of the followed renarrators in the UI to notify the current renarrator.

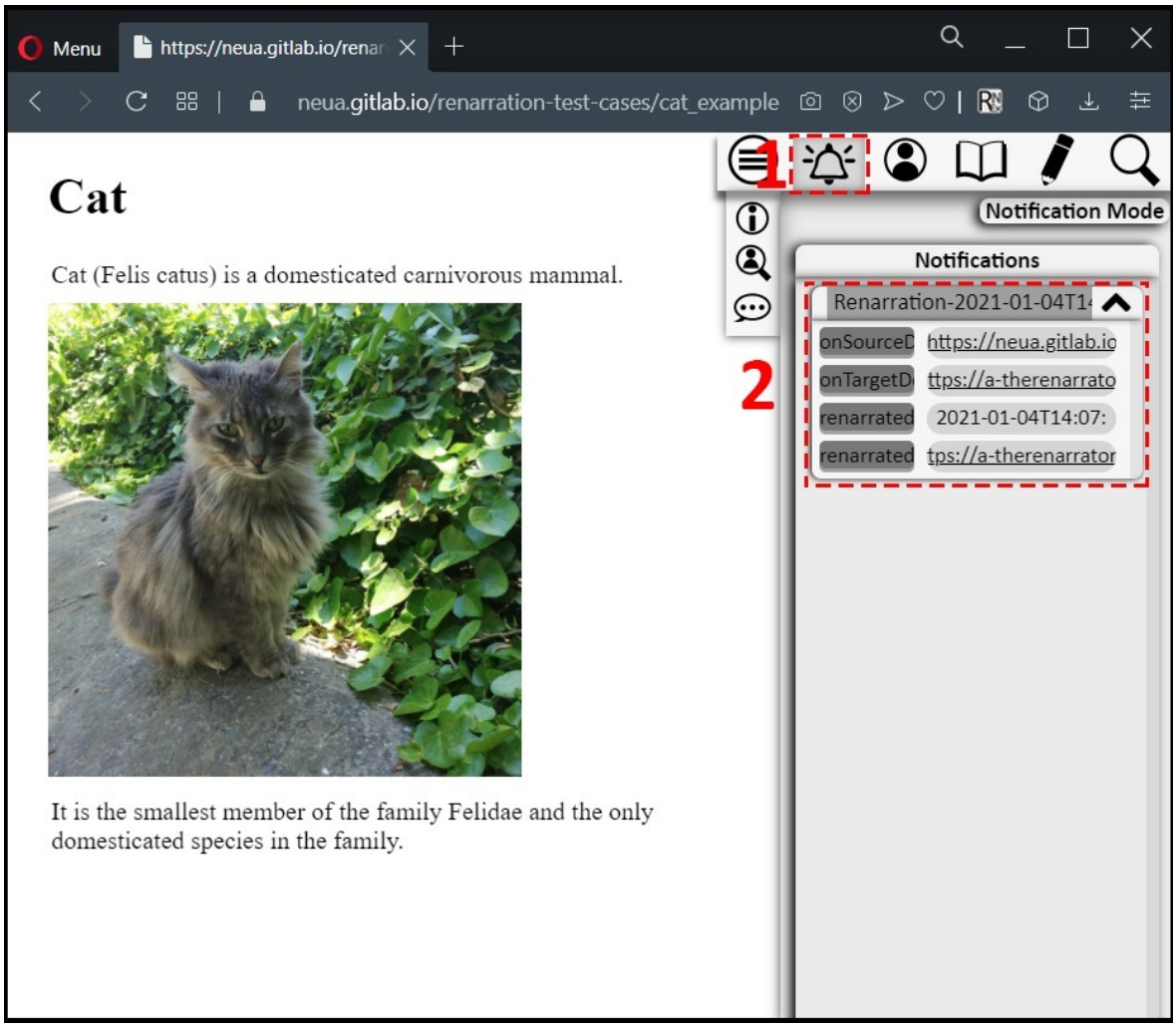

Figure 6.21. Notification functionality of the prototype

6.9. Recommendation Implementation

To get recommendation, the renarrator chooses the icon for the recommendation on RNEx extension UI then can see the list of recommended renarrations. In the Figure 6.22 recommendation functionality of the prototype can be seen.

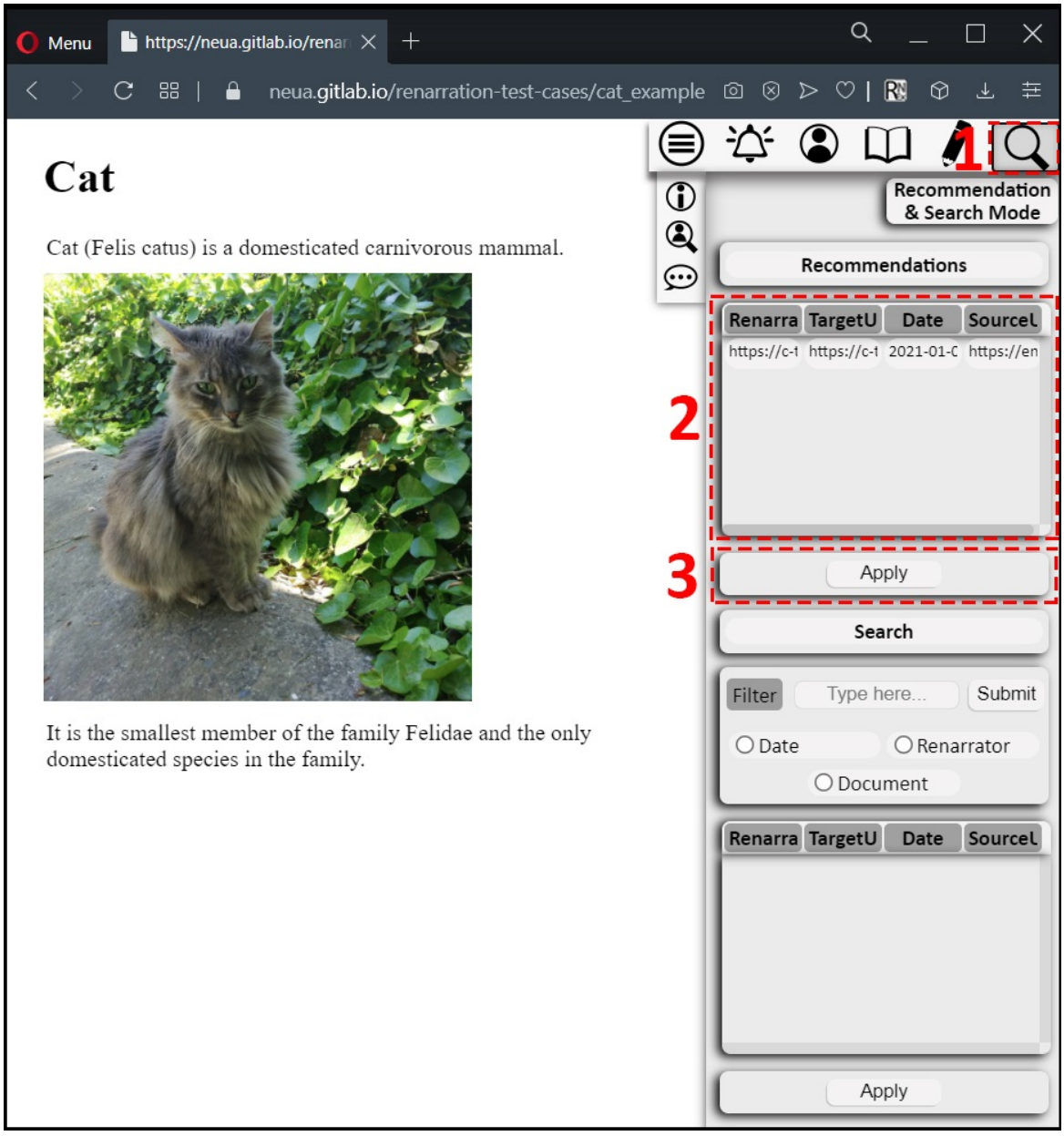

Figure 6.22. Recommendation functionality of the prototype

Numbers near the dotted red boxes in the Figure 6.22 depict the UI of RNEx. Number 1 is button for viewing the recommendations and searching the renarrations.

Number 2 is the box that shows the renarration table and each row corresponds to a renarration, and the table shows the source document URL, location, creation date of the renarration and WebID of its renarrator. Number 3 is button for applying and viewing the selected renarration.

Current recommendation implementation is a basic way of recommending renarrations. Firstly, RNEx retrieves the RenarrationList.ttl and ResponseList.ttl files from the RNEx-Central, then RNEx checks the responses of logged in renarrator and derives the owners of these responded renarrations, then similarly RNEx gets these renarrators' responses and derives owners of responded renarrations, finally RNEx shows the renarrations of the last obtained renarrators as a recommended renarrations in the UI.

6.10. Searching Implementation

To search the renarrations, the renarrator chooses the icon for the searching on RNEx extension UI then can see the list of searched renarrations. In the Figure 6.23 searching functionality of the prototype can be seen.

Numbers near the dotted red boxes in the Figure 6.23 depict the UI of RNEx. Number 1 is button for viewing the recommendations and searching the renarrations. Number 2 is the box that shows the area for searching renarrations depending on a date, renarrator or source document.

Number 3 is the input area for searching renarrations. Number 4 is the renarration table that shows the renarrations that are found for the given search input and criteria, and each row corresponds to a renarration, and the rows show the source document URL, location, creation date of the renarration and WebID of its renarrator. Number 5 is button for applying and viewing the selected renarration.

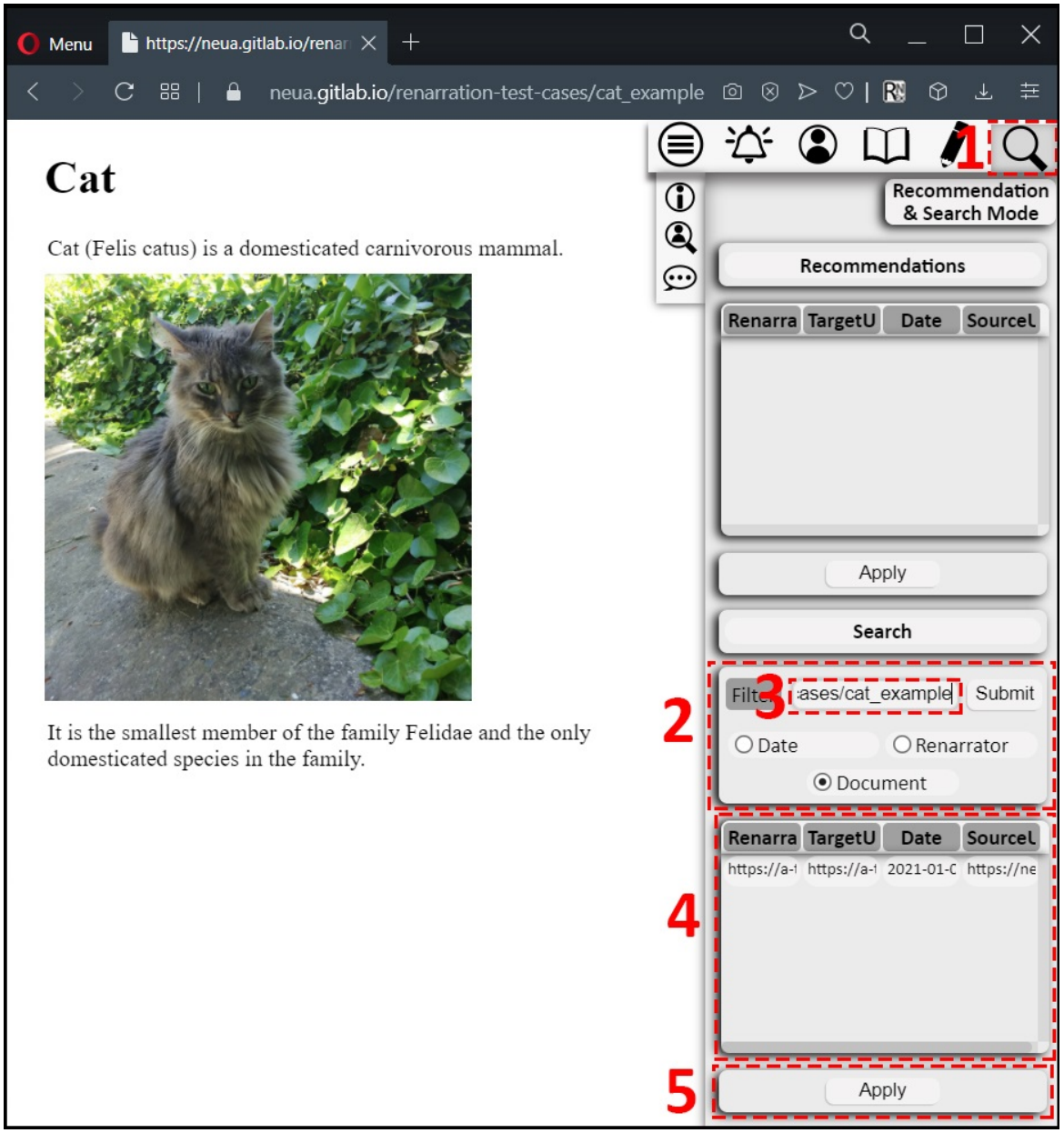

Figure 6.23. Searching functionality of the prototype

In the background, when the renarrator searches renarrations, RNEx checks the RenarrationList.ttl, the Figure 6.24 shows a simple example of the stored information in RenarrationList.ttl, inside the RNEx-Central with SPARQL query in the Figure 6.25 and gets the list of renarrations depending on the chosen criteria. Lastly, RNEx shows the result of the searched renarrations in the UI.

```
1 @prefix c:<https://a-therenarrator.inrupt.net/profile/card#>.
2 @prefix as:<https://www.w3.org/ns/activitystreams#>.
3 @prefix rnsoc:<http://soslab.cmpe.boun.edu.tr/ontologies/rnsoc.owl>.
4 @prefix str:<https://neua.gitlab.io/stray-dogs/>.
5 @prefix RNCNT:<https://a-therenarrator.inrupt.net/renarration/
    RenarrationFiles/RNCNT-kg2eaz3c-13e21d2f-a8e31dda.ttl#>.
6
7 RNCNT:kg2eaz3c-13e21d2f-a8e31dda
8 a rnsoc:Renarration;
9 rnsoc:onSourceDocument str:stray_dogs_example.
10
11 [
12     a as:Create;
13     as:actor c:me;
14     as:object RNCNT:kg2eaz3c-13e21d2f-a8e31dda;
15     as:published "2020-11-09T10:21:13.032Z"^^XML:dateTime
16 ].
```

Figure 6.24. Queried example RenarrationList for searching functionality

```
1 PREFIX rnsoc:<http://soslab.cmpe.boun.edu.tr/ontologies/rnsoc.owl>
2 PREFIX as:<https://www.w3.org/ns/activitystreams#>
3
4 SELECT ?Renarrator  ?PublishDate ?Renarration  ?SourceDocument
5 WHERE {
6  ?blankNode a as:Create;
7             as:object ?Renarration;
8             as:published ?PublishDate;
9             as:actor ?Renarrator.
10
11  ?Renarration rnsoc:onSourceDocument ?SourceDocument.
12 }
```

Figure 6.25. SPARQL query for searching functionality

The RNEx handles SPARQL queries with the rdflib.js. The general SPARQL query for searching is shown in the Figure 6.25. The variables in the query changes depending on the chosen criteria in the UI, i.e., renarrator, date, document.

The query gets the result depending on the logged in renarrator's input. In the Figure 6.26 a query for finding the renarrations of the renarrator "a-therenarrator" (Line:9) can be seen. It is basically the same query in the Figure 6.25 with the desired criteria and input.

```
PREFIX rnsoc:<http://soslab.cmpe.boun.edu.tr/ontologies/rnsoc.owl>
PREFIX as:<https://www.w3.org/ns/activitystreams#>

SELECT ?Renarrator ?PublishDate ?Renarration  ?SourceDocument
WHERE {
 ?blankNode a as:Create;
   as:object ?Renarration;
   as:published ?PublishDate;
   as:actor <https://a-therenarrator.inrupt.net/profile/card#me>.

 ?Renarration rnsoc:onSourceDocument ?SourceDocument.
}
```

Figure 6.26. SPARQL query for searching renarrator

7. EVALUATION

In this chapter we examine the feasiblity of the social renarration platform for specifying renarrations and interactions associated with them. Section 7.1 examines the expressivity of Renarration Social Ontology through numerous test cases. Section 7.2 evaluated the capability of the proposed approach using the RNEx prototype with a detailed use case scenario. Finally, the observations from these evaluations are described in Section 7.3. Throughout this chapter:

- A source document refers to a web document for which a renarrator provides an alternative renarration. Source documents are represented using with HTML.
- A renarration specification refers to a renarration that is represented using the Renarration Social Ontology. All renarration specifications are represented in Turtle syntax.
- A renarrated document refers to a web document that results from applying a renarration specification to a source document. Renarrated documents are represented using with HTML.

7.1. Evaluation of the Proposed Model Using Test Cases

In order to evaluate the proposed model, a set of test cases are created to verify that the model is capable of representing various renarration types that involve the replacement or insertion of an alternative narration and the removal of content.

```
1 @prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
2 @prefix owl: <http://www.w3.org/2002/07/owl#>.
3 @prefix xsd: <http://www.w3.org/2001/XMLSchema#>.
4 @prefix oa: <http://www.w3.org/ns/oa#>.
5 @prefix dc: <http://purl.org/dc/elements/1.1/>.
6 @prefix foaf: <http://xmlns.com/foaf/0.1/>.
7 @prefix rnsoc: <http://soslab.cmpe.boun.edu.tr/ontologies/rnsoc.owl>.
```

Figure 7.1. Prefixes of the namespaces that are used in test cases

The Figure 7.1 shows the common prefixes for the namespaces used in the semantic representation of the test cases. These prefixes are: Resource Description Frame-

work [67] (Line:1), Web Ontology Language [68] (Line:2), XML Schema Definition [69] (Line:3), The Web Annotation Data Model [13] (Line:4), Dublin Core Elements [45] (Line:5), Friend Of A Friend Vocabulary [43] (Line:6) and Renarration Social Ontology [12] (Line:7), respectively.

7.1.1. Replacement of an element with a text element

This test case demonstrates that replacement of an element with a new text element can be described using the proposed model. Consider the source document shown in Figure 7.2 whose body consists of three paragraphs. To specify a renarration where the second paragraph `"<p>Second Text</p>"` to be replaced with `"<p>2nd Text</p>"`, the renarrator must select the source and provide the alternative text.

```
1 <html>
2   <head></head>
3   <body>
4     <p>First Text</p>
5     <p>Second Text</p>
6     <p>Third Text</p>
7   </body>
8 </html>
```

Figure 7.2. A source document that has three paragraphs

Figure 7.3 shows the renarration that specifies this transformation. Line:13 shows the creation time of the renarration in "xsd:dateTime" format. This renarration has a single renarration transformation of type "replace" (Line:24). The target (`"<p>Second Text</p>"`) to be replaced is specified via *oa:XPathSelector* "/html[1]/body[1]/p[2]" (Line:21) which corresponds to the second paragraph in the body.

The alternative narration is specified with *rnsoc:hasNarration* as English text in HTML format whose value is `"<p>2nd Text</p>"` (Line:30). The value here is provided as HTML as the renarrator wants to preserve the paragraph. The source document is specified with *rnsoc:onSourceDocument* and the creator with *rnsoc:renarratedBy* properties.

```
@prefix : <#>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix owl: <http://www.w3.org/2002/07/owl#>.
@prefix rnsoc: <http://soslab.cmpe.boun.edu.tr/ontologies/rnsoc.owl>.
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.
@prefix oa: <http://www.w3.org/ns/oa#>.
@prefix dc: <http://purl.org/dc/elements/1.1/>.
@prefix ex: <http://www.example.com/>.
@prefix c: </profile/card#>.

:kgzmyj1x-9vx3x2sl-sda9z36f
  a owl:NamedIndividual, rnsoc:Renarration;
  rnsoc:createdAt "2020-11-01T21:37:58.005Z"^^xsd:dateTime;
  rnsoc:hasMotivation "alteration"^^xsd:string;
  rnsoc:hasRenarrationTransformation
    [
      a rnsoc:RenarrationTransformation;
      oa:hasSelector
        [
          a oa:XPathSelector;
          rdf:value "/html[1]/body[1]/p[2]"
        ];
      rnsoc:createdAt "2020-11-01T21:37:49.961Z"^^xsd:dateTime;
      rnsoc:hasAction "replace"^^xsd:string;
      rnsoc:hasNarration
        [
          a rnsoc:Text;
          dc:format "text/html";
          dc:language "en";
          rdf:value "<p>2nd Text</p>"
        ]
    ];
    rnsoc:onSourceDocument ex:source_document;
    rnsoc:renarratedBy c:me.
  ex:source_document a rnsoc:Document.
```

Figure 7.3. A renarration specification with a single transformation that specifies with a single transformation that specifies the replacement of the second paragraph with a paragraph

When this renarration is applied to the the source document, which would happen when a user is viewing the source document and requests to see this renarration, the transformations will be successively applied. In this case the resulting renarration will be as shown in the Figure 7.4. The change in the source document after the renarration process is in the Line:5 of the renarrated document.

```
1 <html>
2   <head></head>
3   <body>
4     <p>First Text</p>
5     <p>2nd Text</p>
6     <p>Third Text</p>
7   </body>
8 </html>
```

Figure 7.4. A renarrated document obtained by applying the renarration specification shown in Figure 7.3 on the source document shown in Figure 7.2

7.1.2. Replacement of an element with an audio element

This test case demonstrates that replacement of an element with an audio element can be described using the proposed Renarration Social Ontology. Accordingly, the `"<p>Second Text</p>"` inside the source document shown in the Figure 7.2, is replaced with an audio element. The renarration in the Figure 7.3 is same for renarration of this case, thus only difference in the line between Line:25 and Line:31 of Figure 7.3 is demonstrated in the Figure 7.5

```
1 rnsoc:hasNarration
2   [
3     a rnsoc:Audio;
4     dc:format "text/html";
5     dc:language "en";
6     rdf:value
7       "<audio controls>
8          <source src="http://www.example.com/audio.mp3" type="
      audio/mpeg">
9        </audio>"
10  ]
```

Figure 7.5. A renarration specification with a single transformation that specifies the replacement of the second paragraph with an audio

In the Figure 7.6 source document after the renarration process, namely renarrated document, can be observed. The change in the source document (Figure 7.2)

after the renarration process (Figure 7.5) is between the Line:5 and the Line:7 of the renarrated document in the Figure 7.6.

```
1 <html>
2   <head></head>
3   <body>
4     <p>First Text</p>
5     <audio controls>
6       <source src="http://www.example.com/audio.mp3" type="audio/
     mpeg">
7     </audio>
8     <p>Third Text</p>
9   </body>
10 </html>
```

Figure 7.6. A renarrated document obtained by applying the renarration specification shown in Figure 7.5 on the source document shown in Figure 7.2

7.1.3. Replacement of an element with an image element

This test case demonstrates that replacement of an element with an image element can be described using the proposed Renarration Social Ontology. Accordingly, the "`<p>Second Text</p>`" inside the source document shown in the Figure 7.2, is replaced with an image element. The renarration in the Figure 7.3 is same for renarration of this case, thus only difference in the line between Line:25 and Line:31 of Figure 7.3 is shown in the Figure 7.7

```
1 rnsoc:hasNarration
2   [
3     a rnsoc:Image;
4     dc:format "text/html";
5     dc:language "en";
6     rdf:value "<img src="http://www.example.com/image.png">"
7   ]
```

Figure 7.7. A renarration specification with a single transformation that specifies the replacement of the second paragraph with an image

In the Figure 7.8 source document after the renarration process, namely renarrated document, can be observed. The change in the source document (Figure 7.2) after the renarration process (Figure 7.7) is in the Line:5 of the renarrated document in the Figure 7.8.

```
1 <html>
2   <head></head>
3   <body>
4     <p>First Text</p>
5     <img src="http://www.example.com/image.png">
6     <p>Third Text</p>
7   </body>
8 </html>
```

Figure 7.8. A renarrated document obtained by applying the renarration specification shown in Figure 7.7 on the source document shown in Figure 7.2

7.1.4. Replacement of an element with a video element

This test case demonstrates that replacement of an element with a video element can be described using the proposed Renarration Social Ontology. Accordingly, the `"<p >Second Text</p>"` inside the source document shown in the Figure 7.2, is replaced with a video element. The renarration specification in the Figure 7.3 is same for renarration specification of this case, thus only difference in the line between Line:25 and Line:31 of Figure 7.3 is displayed in the Figure 7.9

```
1 rnsoc:hasNarration
2   [
3     a rnsoc:Video;
4     dc:format "text/html";
5     dc:language "en";
6     rdf:value
7      "<video controls>
8         <source src="http://www.example.com/video.mp4" type="video
     /mp4">
9       </video>"
10  ]
```

Figure 7.9. A renarration specification with a single transformation that specifies the replacement of the second paragraph with a video

In the Figure 7.10 source document after the renarration process, namely renarrated document, can be observed. The change in the source document (Figure 7.2) after the renarration process (Figure 7.9) is between the Line:5 and the Line:7 of the renarrated document in the Figure 7.10.

```
 1 <html>
 2   <head></head>
 3   <body>
 4     <p>First Text</p>
 5     <video controls>
 6       <source src="http://www.example.com/video.mp4" type="video/
      mp4">
 7     </video>
 8     <p>Third Text</p>
 9   </body>
10 </html>
```

Figure 7.10. A renarrated document obtained by applying the renarration specification shown in Figure 7.9 on the source document shown in Figure 7.2

7.1.5. Removal of an element

This test case demonstrates that removal of an element can be described using the proposed model. For this purpose, `"<p>First Text</p>"` in the Figure 7.2 is removed. The Line:19 of the renarration specification (Figure 7.11) indicates the XPath of the chosen `"<p>First Text</p>"` that will be removed (Line:22).

```
 1 @prefix : <#>.
 2 @prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
 3 @prefix owl: <http://www.w3.org/2002/07/owl#>.
 4 @prefix rnsoc: <http://soslab.cmpe.boun.edu.tr/ontologies/rnsoc.owl>.
 5 @prefix xsd: <http://www.w3.org/2001/XMLSchema#>.
 6 @prefix oa: <http://www.w3.org/ns/oa#>.
 7 @prefix ex: <http://www.example.com/>.
 8 @prefix c: </profile/card#>.
 9
10 :kgzpkkxj-anlsfat6-3cvs5m7e
11   a owl:NamedIndividual, rnsoc:Renarration;
12   rnsoc:createdAt "2020-11-01T22:51:06.103Z"^^xsd:dateTime;
13   rnsoc:hasMotivation "alteration"^^xsd:string;
14   rnsoc:hasRenarrationTransformation
15     [
16       oa:hasSelector
17       [
18         a oa:XPathSelector;
19         rdf:value "/html[1]/body[1]/p[1]"
20       ];
21       rnsoc:createdAt "2020-11-01T22:51:04.460Z"^^xsd:dateTime;
22       rnsoc:hasAction "remove"^^xsd:string
23     ];
24   rnsoc:onSourceDocument ex:source_document;
25   rnsoc:renarratedBy c:me.
26 ex:source_document a rnsoc:Document.
```

Figure 7.11. A renarration specification with a single transformation that specifies the removal of the first paragraph

In the Figure 7.12 source document after the renarration process, namely renarrated document, can be observed. Moreover, it can be seen that renarrated document doesn't contain `"<p>First Text</p>"` because of the removal operation.

```
1 <html>
2   <head></head>
3   <body>
4     <p>Second Text</p>
5     <p>Third Text</p>
6   </body>
7 </html>
```

Figure 7.12. A renarrated document obtained by applying the renarration specification shown in Figure 7.11 on the source document shown in Figure 7.2

7.1.6. Insertion of a new element between two elements

This test case demonstrates the insertion of the element `"<p>1-2 Text</p>"` between the element `"<h1>First Text</h1>"` and the element `"<h1>Second Text</h1>"` in the source document shown in Figure 7.2. See Section 5.1.2 for the description of insertion points. The Line:24 of the renarration specification (Figure 7.14) indicates the XPath of the start selector. The Line:29 of the renarration specification (Figure 7.14) indicates the XPath of the end selector. The renarration specification indicates that `"<h1>1-2 Text</h1>"` (Line:39) should be inserted (Line:33).

In the Figure 7.13 renarrated document can be observed. The change in the source document after the renarration process is in the Line:5 of the renarrated document.

```
1 <html>
2   <head></head>
3   <body>
4     <p>First Text</p>
5     <p>1-2 Text</p>
6     <p>Second Text</p>
7     <p>Third Text</p>
8   </body>
9 </html>
```

Figure 7.13. A renarrated document obtained by applying the renarration specification shown in Figure 7.14 on the source document shown in Figure 7.2

```
@prefix : <#>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix owl: <http://www.w3.org/2002/07/owl#>.
@prefix rnsoc: <http://soslab.cmpe.boun.edu.tr/ontologies/rnsoc.owl>.
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.
@prefix oa: <http://www.w3.org/ns/oa#>.
@prefix dc: <http://purl.org/dc/elements/1.1/>.
@prefix ex: <http://www.example.com/>.
@prefix c: </profile/card#>.

:kh0l6iw2-jlhr884y-u93ef9na
  a owl:NamedIndividual, rnsoc:Renarration;
  rnsoc:createdAt "2020-11-02T13:35:57.986Z"^^xsd:dateTime;
  rnsoc:hasMotivation "alteration"^^xsd:string;
  rnsoc:hasRenarrationTransformation
    [
      a rnsoc:RenarrationTransformation;
      oa:hasSelector
        [
          a oa:RangeSelector;
          oa:hasStartSelector
              [
                a oa:XPathSelector;
                rdf:value "/html[1]/body[1]/p[1]"
              ];
          oa:hasEndSelector
              [
                a oa:XPathSelector;
                rdf:value "/html[1]/body[1]/p[2]"
              ]
        ];
      rnsoc:createdAt "2020-11-02T13:35:55.306Z"^^xsd:dateTime;
      rnsoc:hasAction "insert"^^xsd:string;
      rnsoc:hasNarration
        [
          a rnsoc:Text;
          dc:format "text/html";
          dc:language "en";
          rdf:value "<p>1-2 Text</p>"
        ]
    ];
  rnsoc:onSourceDocument ex:source_document;
  rnsoc:renarratedBy c:me.
ex:source_document a rnsoc:Document.
```

Figure 7.14. A renarration specification with a single transformation that specifies the insertion of a paragraph between two paragraphs

7.1.7. Insertion of a new element at the beginning of a sequence

This test case demonstrates the insertion of the element `"<h1>Article</h1>"` before the element `"<h1>First Heading</h1>"` in the source document shown in Figure 7.15. The start selector Line:23 doesn't have a XPath as it indicates the beginning of a sequence (relative to the end selector which is a `"<div>"` element). The Line:30 of the renarration specification (Figure 7.17) indicates the XPath of the end selector. The renarration specification indicates that `"<h1>Article</h1>"` (Line:41) should be inserted (Line:34).

```
 1 <html>
 2 <head></head>
 3 <body>
 4   <div>
 5     <h1>First Heading</h1>
 6     <p>First Text</p>
 7   </div>
 8   <div>
 9     <h1>Second Heading</h1>
10     <p>Second Text</p>
11   </div>
12 </body>
13 </html>
```

Figure 7.15. A source document that has two headings and two paragraphs

In the Figure 7.16 renarrated document can be observed. The change in the source document after the renarration process is in the Line:5 of the renarrated document.

```
 1 <html>
 2 <head></head>
 3 <body>
 4   <div>
 5     <h1>Article</h1>
 6     <h1>First Heading</h1>
 7     <p>First Text</p>
 8   </div>
 9   <div>
10     <h1>Second Heading</h1>
11     <p>Second Text</p>
12   </div>
13 </body>
14 </html>
```

Figure 7.16. A renarrated document obtained by applying the renarration specification shown in Figure 7.17 on the source document shown in Figure 7.15

```
@prefix : <#>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix owl: <http://www.w3.org/2002/07/owl#>.
@prefix rnsoc: <http://soslab.cmpe.boun.edu.tr/ontologies/rnsoc.owl>.
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.
@prefix oa: <http://www.w3.org/ns/oa#>.
@prefix dc: <http://purl.org/dc/elements/1.1/>.
@prefix ex: <http://www.example.com/>.
@prefix c: </profile/card#>.

:kif1xfvb-5carfxhf-b6ah8r5j
  a owl:NamedIndividual, rnsoc:Renarration;
  rnsoc:createdAt "2020-12-07T21:13:16.439Z"^^xsd:dateTime;
  rnsoc:hasMotivation "alteration"^^xsd:string;
  rnsoc:hasRenarrationTransformation
    [
      a rnsoc:RenarrationTransformation;
      oa:hasSelector
        [
          a rnsoc:RangeSelector;
          oa:hasStartSelector
              [
                a oa:XPathSelector ;
                rdf:value ""
              ];
          oa:hasEndSelector
              [
                a oa:XPathSelector;
                rdf:value
                "/html[1]/body[1]/div[1]/h1[1]"
              ]
        ];
      rnsoc:createdAt "2020-12-07T21:13:12.187Z"^^xsd:dateTime;
      rnsoc:hasAction "insert"^^xsd:string;
      rnsoc:hasNarration
          [
            a rnsoc:Text;
            dc:format "text/html";
            dc:language "en";
            rdf:value
                "<h1>Article</h1>"
          ]
    ];
  rnsoc:onSourceDocument ex:source_document;
  rnsoc:renarratedBy c:me.
ex:source_document a rnsoc:Document.
```

Figure 7.17. A renarration specification with a single transformation that specifies the insertion of a heading before the first heading

7.1.8. Insertion of a new element at the end of the sequence

This test case demonstrates the insertion of the element `"<p>Third Text</p>"` after the element `"<p>Second Text</p>"` in the source document shown in Figure 7.15.

```
@prefix : <#>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix owl: <http://www.w3.org/2002/07/owl#>.
@prefix rnsoc: <http://soslab.cmpe.boun.edu.tr/ontologies/rnsoc.owl>.
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.
@prefix oa: <http://www.w3.org/ns/oa#>.
@prefix dc: <http://purl.org/dc/elements/1.1/>.
@prefix ex: <http://www.example.com/>.
@prefix c: </profile/card#>.

:kifrhnpm-bs76jy6j-axzrhwju
  a owl:NamedIndividual, rnsoc:Renarration;
  rnsoc:createdAt "2020-12-08T09:08:50.122Z"^^xsd:dateTime;
  rnsoc:hasMotivation "alteration"^^xsd:string;
  rnsoc:hasRenarrationTransformation
    [
      a rnsoc:RenarrationTransformation;
      oa:hasSelector
        [
          a rnsoc:RangeSelector;
            oa:hasStartSelector
              [
                a oa:XPathSelector;
                rdf:value
                "/html[1]/body[1]/div[2]/p[1]"
              ];
            oa:hasEndSelector
              [
                a oa:XPathSelector ;
                rdf:value ""
              ]
        ];
      rnsoc:createdAt "2020-12-08T09:08:42.808Z"^^xsd:dateTime;
      rnsoc:hasAction "insert"^^xsd:string;
      rnsoc:hasNarration
          [
            a rnsoc:Text;
            dc:format "text/html";
            dc:language "en";
            rdf:value
                "<p>Third Text</p>"
          ]
    ];
  rnsoc:onSourceDocument ex:source_document;
  rnsoc:renarratedBy c:me.
ex:source_document a rnsoc:Document.
```

Figure 7.18. A renarration specification with a single transformation that specifies the insertion of a paragraph after the last paragraph

The end selector Line:29 doesn't have a XPath as it indicates the end of a sequence (relative to the end selector which is a `"<div>"` element). The Line:25 of the renarration specification (Figure 7.18) indicates the XPath of the start selector. The renarration specification indicates that `"<p>Third Text</p>"` (*rnsoc:hasNarration* property Line:41) should be inserted (Line:34) between the above mentioned elements.

In the Figure 7.19 renarrated document can be observed. The change in the source document after the renarration process is in the Line:11 of the renarrated document.

```
 1 <html>
 2 <head></head>
 3 <body>
 4   <div>
 5     <h1>First Heading</h1>
 6     <p>First Text</p>
 7   </div>
 8   <div>
 9     <h1>Second Heading</h1>
10     <p>Second Text</p>
11     <p>Third Text</p>
12   </div>
13 </body>
14 </html>
```

Figure 7.19. A renarrated document obtained by applying the renarration specification shown in Figure 7.18 on the source document shown in Figure 7.15

7.1.9. Replacement of text characters

This test case demonstrates that replacement of text characters can be described using the proposed model. Accordingly, "Second" word inside the `"<p>Second Text</p>"` HTML element, is replaced with "2nd" word. Source document can be seen in the Figure 7.2.

The Line:25 and the Line:26 of the renarration specification (Figure 7.20) indicates the start and end characters of the text selection, respectively. Moreover, renarration specification indicates that value "2nd" in the Line:36 will be replaced (Line:30) with the selected text in the source document after the renarration process. In the Figure 7.21 renarrated document can be observed. The change in the source document after the renarration process is in the Line:5 o f the renarrated document.

```
@prefix : <#>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix owl: <http://www.w3.org/2002/07/owl#>.
@prefix rnsoc: <http://soslab.cmpe.boun.edu.tr/ontologies/rnsoc.owl>.
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.
@prefix oa: <http://www.w3.org/ns/oa#>.
@prefix dc: <http://purl.org/dc/elements/1.1/>.
@prefix ex: <http://www.example.com/>.
@prefix c: </profile/card#>.

:kezlyg1x-8uc3x2sl-cdb9a26f
  a owl:NamedIndividual, rnsoc:Renarration;
  rnsoc:createdAt "2020-11-01T21:37:58.005Z"^^xsd:dateTime;
  rnsoc:hasMotivation "alteration"^^xsd:string;
  rnsoc:hasRenarrationTransformation
    [
      a rnsoc:RenarrationTransformation;
      oa:hasSelector
        [
          a oa:XPathSelector;
          rdf:value "/html[1]/body[1]/p[2]";
          oa:refinedBy
            [
              a oa:TextPositionSelector;
              oa:start "0";
              oa:end "6";
            ];
        ];
      rnsoc:createdAt "2020-11-01T21:37:49.961Z"^^xsd:dateTime;
      rnsoc:hasAction "replace"^^xsd:string;
      rnsoc:hasNarration
        [
          a rnsoc:Text;
          dc:format "text/plain";
          dc:language "en";
          rdf:value "2nd"
        ]
    ];
  rnsoc:onSourceDocument ex:source_document;
  rnsoc:renarratedBy c:me.
ex:source_document a rnsoc:Document.
```

Figure 7.20. A renarration specification with a single transformation that specifies the replacement of text characters

```
<html>
  <head></head>
  <body>
    <p>First Text</p>
    <p>2nd Text</p>
    <p>Third Text</p>
  </body>
</html>
```

Figure 7.21. A renarrated document obtained by applying the renarration specification shown in Figure 7.20 on the source document shown in Figure 7.2

7.1.10. Removal of text characters

This test case demonstrates that removal of text characters can be described using the proposed model. Accordingly, "Lemon" word inside the `"<p>Lemon Banana Strawberry</p>"` HTML element, is removed. Source document can be seen in the Figure 7.22.

```
<html>
  <head></head>
  <body>
    <p>Lemon Banana Strawberry</p>
  </body>
</html>
```

Figure 7.22. A source document that has a single paragraph

```
@prefix : <#>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix owl: <http://www.w3.org/2002/07/owl#>.
@prefix rnsoc: <http://soslab.cmpe.boun.edu.tr/ontologies/rnsoc.owl>.
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.
@prefix oa: <http://www.w3.org/ns/oa#>.
@prefix dc: <http://purl.org/dc/elements/1.1/>.
@prefix ex: <http://www.example.com/>.
@prefix c: </profile/card#>.

:kezlyg1x-8uc3x2sl-cdb9a26f
  a owl:NamedIndividual, rnsoc:Renarration;
  rnsoc:createdAt "2020-11-01T21:37:58.005Z"^^xsd:dateTime;
  rnsoc:hasMotivation "alteration"^^xsd:string;
  rnsoc:hasRenarrationTransformation
    [
      a rnsoc:RenarrationTransformation;
      oa:hasSelector
        [
          a oa:XPathSelector;
          rdf:value "/html[1]/body[1]/p[1]";
          oa:refinedBy
            [
              a oa:TextPositionSelector;
              oa:start "0";
              oa:end "6";
            ];
        ];
      rnsoc:createdAt "2020-11-01T21:37:49.961Z"^^xsd:dateTime;
      rnsoc:hasAction "remove"^^xsd:string;
    ];
  rnsoc:onSourceDocument ex:source_document;
  rnsoc:renarratedBy c:me.
ex:source_document a rnsoc:Document.
```

Figure 7.23. A renarration specification with a single transformation that specifies the removal of text characters

In the Figure 7.23 renarration specification of the removal of text characters example can be seen. The Line:25 and the Line:26 of the renarration specification (Figure 7.23) indicates the start and end characters of the text selection, respectively. Moreover, renarration specification indicates that selected value will be removed (Line:30) from the source document after the renarration process.

In the Figure 7.24 renarrated document can be observed. The change in the source document after the renarration process is in the Line:4 of the renarrated document.

```
1 <html>
2   <head></head>
3   <body>
4     <p>Banana Strawberry</p>
5   </body>
6 </html>
```

Figure 7.24. A renarrated document obtained by applying the renarration specification shown in Figure 7.23 on the source document shown in Figure 7.22

7.1.11. Insertion of text characters

This test case demonstrates that insertion of text characters can be described using the proposed model. Accordingly, after the "Banana" word inside the `"<p>Lemon Banana Strawberry</p>"` HTML element, "Pear" word is inserted. Source document can be seen in the Figure 7.22.

In the Figure 7.25 renarration specification of the insertion of text characters example can be seen. The Line:25 and the Line:26 of the renarration (Figure 7.25) indicates the start and end characters of the text selection, respectively. Moreover, renarration specification indicates that value "Pear" in the Line:36 will be inserted (Line:30) with the selected text in the source document after the renarration process.

In the Figure 7.26 renarrated document can be observed. The change in the source document after the renarration process is in the Line:4 of the renarrated document.

```
@prefix : <#>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix owl: <http://www.w3.org/2002/07/owl#>.
@prefix rnsoc: <http://soslab.cmpe.boun.edu.tr/ontologies/rnsoc.owl>.
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.
@prefix oa: <http://www.w3.org/ns/oa#>.
@prefix dc: <http://purl.org/dc/elements/1.1/>.
@prefix ex: <http://www.example.com/>.
@prefix c: </profile/card#>.

:kezlyg1x-8uc3x2sl-cdb9a26f
  a owl:NamedIndividual, rnsoc:Renarration;
  rnsoc:createdAt "2020-11-01T21:37:58.005Z"^^xsd:dateTime;
  rnsoc:hasMotivation "alteration"^^xsd:string;
  rnsoc:hasRenarrationTransformation
    [
      a rnsoc:RenarrationTransformation;
      oa:hasSelector
        [
          a oa:XPathSelector;
          rdf:value "/html[1]/body[1]/p[1]";
          oa:refinedBy
            [
              a oa:TextPositionSelector;
              oa:start "13";
              oa:end "13";
            ];
        ];
      rnsoc:createdAt "2020-11-01T21:37:49.961Z"^^xsd:dateTime;
      rnsoc:hasAction "insert"^^xsd:string;
      rnsoc:hasNarration
        [
          a rnsoc:Text;
          dc:format "text/plain";
          dc:language "en";
          rdf:value "Pear "
        ]
    ];
  rnsoc:onSourceDocument ex:source_document;
  rnsoc:renarratedBy c:me.
ex:source_document a rnsoc:Document.
```

Figure 7.25. A renarration specification with a single transformation that specifies the insertion of text characters

```
<html>
  <head></head>
  <body>
    <p>Lemon Banana Pear Strawberry</p>
  </body>
</html>
```

Figure 7.26. A renarrated document obtained by applying the renarration specification shown in Figure 7.25 on the source document shown in Figure 7.22

7.1.12. Complex operations using various actions and narration types

This test case demonstrates that complex replace, remove, insert operations using text, audio, image elements can be described using the proposed model. To demonstrate it renarration specification with four renarration transformations is applied to Figure 7.27 source document. In the first renarration transformation, "Second Text" content is replaced with an image. In the second renarration transformation, "First Text" content is removed. In the third renarration transformation, "First Heading" content is replaced with a "1st Heading" text. In the fourth renarration transformation, between the newly added image and "Third Text" an audio element is inserted.

```
 1 <html>
 2   <head></head>
 3   <body>
 4     <h1>First Heading</h1>
 5     <p>First Text</p>
 6     <div>
 7       <h1>Second Heading</h1>
 8       <p>Second Text</p>
 9       <p>Third Text</p>
10     </div>
11   </body>
12 </html>
```

Figure 7.27. A source document that has two headings and three paragraphs

Figure 7.28 and Figure 7.29 are the parts of a single renarration specification. Figure 7.28 shows the renarration specification of the complex operations using various actions and narration types example first part. First renarration transformation between the Line:16 and the Line:33 has the replace action (Line:24) and `"<img src ="http://www.example.com/image.png">"` narration (Line:31). Second renarration transformation between the Line:34 and the Line:43 has the remove action (Line:42) and no narration since it is a removal operation.

Figure 7.29 shows the renarration specification of the complex operations using various actions and narration types example second part. Third renarration transformation between the Line:1 and the Line:18 has the replace action (Line:9) and `"<p >1st Heading</p>"` narration (Line:16). Fourth renarration transformation between the Line:19 and the Line:50 has the insert action (Line:39) and `"<audio controls><source`

`src="http://www.example.com/audio.mp3" type="audio/mpeg"></audio>"` narration (Line:46 - Line:48).

In the Figure 7.30 source document after the renarration process, namely renarrated document, can be observed. The changes in the source document after the renarration process that has four renarration transformation can be seen in the renarrated document.

```
@prefix : <#>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix owl: <http://www.w3.org/2002/07/owl#>.
@prefix rnsoc: <http://soslab.cmpe.boun.edu.tr/ontologies/rnsoc.owl>.
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.
@prefix oa: <http://www.w3.org/ns/oa#>.
@prefix dc: <http://purl.org/dc/elements/1.1/>.
@prefix ex: <http://www.example.com/>.
@prefix c: </profile/card#>.

:khaloqs1-idcbvqc3-2tcbc0z2
  a owl:NamedIndividual, rnsoc:Renarration;
  rnsoc:createdAt "2020-11-09T13:47:49.777Z"^^xsd:dateTime;
  rnsoc:hasMotivation "elaboration"^^xsd:string;
  rnsoc:hasRenarrationTransformation
    [
      a rnsoc:RenarrationTransformation;
      oa:hasSelector
        [
          a oa:XPathSelector;
          rdf:value "/html[1]/body[1]/div[1]/p[1]"
        ];
      rnsoc:createdAt "2020-11-09T13:46:26.666Z"^^xsd:dateTime;
      rnsoc:hasAction "replace"^^xsd:string;
      rnsoc:hasNarration
        [
          a rnsoc:Text;
          dc:format "text/html";
          dc:language "en";
          rdf:value
           "<img  src="http://www.example.com/image.png">"
        ]
    ],
    [
      a rnsoc:RenarrationTransformation;
      oa:hasSelector
        [
          a oa:XPathSelector;
          rdf:value "/html[1]/body[1]/p[1]"
        ];
      rnsoc:createdAt "2020-11-09T13:46:31.527Z"^^xsd:dateTime;
      rnsoc:hasAction "remove"^^xsd:string
    ],
```

Figure 7.28. A renarration specification with four transformations that specifies the complex operations using various actions and narration types first part

```
1     [
2       a rnsoc:RenarrationTransformation;
3       oa:hasSelector
4       [
5         a oa:XPathSelector;
6         rdf:value "/html[1]/body[1]/h1[1]"
7       ];
8       rnsoc:createdAt "2020-11-09T13:46:42.053Z"^^xsd:dateTime;
9       rnsoc:hasAction "replace"^^xsd:string;
10      rnsoc:hasNarration
11        [
12          a rnsoc:Text;
13          dc:format "text/html";
14          dc:language "en";
15          rdf:value
16              "<p>1st Heading</p>"
17        ]
18    ],
19    [
20      a rnsoc:RenarrationTransformation;
21      oa:hasSelector
22        [
23          a rnsoc:RangeSelector;
24          oa:hasStartSelector
25                  [
26                      a oa:XPathSelector;
27                      rdf:value
28                      "/html[1]/body[1]/div[1]/img[1]"
29                  ];
30          oa:hasEndSelector
31                  [
32                      a oa:XPathSelector;
33                      rdf:value
34                      "/html[1]/body[1]/div[1]/p[1]"
35                  ]
36
37        ];
38      rnsoc:createdAt "2020-11-09T13:47:39.681Z"^^xsd:dateTime;
39      rnsoc:hasAction "insert"^^xsd:string;
40      rnsoc:hasNarration
41        [
42          a rnsoc:Audio;
43          dc:format "text/html";
44          dc:language "en";
45          rdf:value
46           "<audio controls>
47              <source src="http://www.example.com/audio.mp3" type=
      "audio/mpeg">
48            </audio>"
49        ]
50      ];
51     rnsoc:onSourceDocument ex:source_document;
52     rnsoc:renarratedBy c:me.
53 ex:source_document a rnsoc:Document.
```

Figure 7.29. A renarration specification with four transformations that specifies the complex operations using various actions and narration types second part

```
 1 <html>
 2   <head></head>
 3   <body>
 4     <h1>1st Heading</h1>
 5     <div>
 6       <h1>Second Heading</h1>
 7       <img src="http://www.example.com/image.png">
 8       <audio controls>
 9         <source src="http://www.example.com/audio.mp3" type="audio
      /mpeg">
10       </audio>
11       <p>Third Text</p>
12     </div>
13   </body>
14 </html>
```

Figure 7.30. A renarrated document obtained by applying the renarration specification shown in Figure 7.28 on the source document shown in Figure 7.27

7.1.13. Response to a renarration with a rate

This test case demonstrates that rating a renarration can be defined using the proposed model. The Figure 7.32 shows that the renarration in the Line:16 rated with the value of (Line:13) by the renarrator in the Line:14.

```
 1 @prefix : <#>.
 2 @prefix c: <https://a-therenarrator.example.com/profile/card#>.
 3 @prefix foaf: <http://xmlns.com/foaf/0.1/>.
 4 @prefix rnsoc: <http://soslab.cmpe.boun.edu.tr/ontologies/rnsoc.owl>.
 5 @prefix xsd: <http://www.w3.org/2001/XMLSchema#>.
 6 @prefix RNC: <https://b-therenarrator.example.com/renarration/
     RenarrationFiles/RNCNT-kgd3ea21-4b21324a-6el3bda3.ttl#>.
 7 @prefix owl: <http://www.w3.org/2002/07/owl#>.
 8
 9 c:me a foaf:Person, rnsoc:Renarrator.
10
11 :kgc2iew2-e7nc99sq-m9phna1f
12     a rnsoc:Rate;
13     rnsoc:hasRateValue "5"^^xsd:int;
14     rnsoc:isCreatedBy c:me;
15     rnsoc:respondedAt "2020-11-09T19:25:14.723Z"^^xsd:dateTime;
16     rnsoc:responseOnRenarration RNC:kgd3ea21-4b21324a-6el3bda3.
17
18 RNC:kgd3ea21-4b21324a-6el3bda3 a owl:NamedIndividual,
19 rnsoc:Renarration.
```

Figure 7.31. Response to a renarration with a rate

7.1.14. Response to a renarration with a comment

This test case demonstrates that commenting a renarration can be described using the proposed model. The Figure 7.32 shows that the renarration in the Line:16 commented with the value of "Some comment." (Line:13) by the renarrator in the Line:14.

```
1 @prefix : <#>.
2 @prefix c: <https://a-therenarrator.example.com/profile/card#>.
3 @prefix foaf: <http://xmlns.com/foaf/0.1/>.
4 @prefix rnsoc: <http://soslab.cmpe.boun.edu.tr/ontologies/rnsoc.owl>.
5 @prefix xsd: <http://www.w3.org/2001/XMLSchema#>.
6 @prefix RNC: <https://b-therenarrator.example.com/renarration/
     RenarrationFiles/RNCNT-kge3bd1a-4a31b23a-7ek3dea2.ttl#>.
7 @prefix owl: <http://www.w3.org/2002/07/owl#>.
8
9     c:me a foaf:Person, rnsoc:Renarrator.
10
11 :kge1a32b-e33edce2-4ea2detc
12     a rnsoc:Comment;
13     rnsoc:hasCommentValue "Some comment."^^xsd:string;
14     rnsoc:isCreatedBy c:me;
15     rnsoc:respondedAt "2020-11-11T09:32:17.143Z"^^xsd:dateTime;
16     rnsoc:responseOnRenarration RNC:kge3bd1a-4a31b23a-7ek3dea2.
17
18 RNC:kge3bd1a-4a31b23a-7ek3dea2 a owl:NamedIndividual,
19 rnsoc:Renarration.
```

Figure 7.32. Response to a renarration with a comment

7.1.15. Referencing a part of an external document

This test case demonstrates that using part of an external document in a renarration can be defined using the proposed model. Accordingly, the `"<p>Third Text</p>"` element in the source document (Figure 7.2) replaced with the `"<p>External Third Paragraph</p>"` element in the external document (Figure 7.33) through a reference to the external document and the XPath. The Figure 7.34 and Figure 7.35 shows the renarration specification and renarrated document, respectively. The part of the external document that is used in the renarration can be seen in the Line:6 of the Figure 7.33.

```
<html>
  <head></head>
  <body>
    <p>External First Paragraph</p>
    <p>External Paragraph</p>
    <p>External Third Paragraph</p>
  </body>
</html>
```

Figure 7.33. A web document whose element is used to specify a renarration transformation

```
@prefix : <#>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix owl: <http://www.w3.org/2002/07/owl#>.
@prefix rnsoc: <http://soslab.cmpe.boun.edu.tr/ontologies/rnsoc.owl>.
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.
@prefix oa: <http://www.w3.org/ns/oa#>.
@prefix dc: <http://purl.org/dc/elements/1.1/>.
@prefix ex: <http://www.example.com/>.
@prefix c: </profile/card#>.

:kgavcj2d-3va1e4s3-aca5b32d
  a owl:NamedIndividual, rnsoc:Renarration;
  rnsoc:createdAt "2020-12-05T11:12:18.005Z"^^xsd:dateTime;
  rnsoc:hasMotivation "alteration"^^xsd:string;
  rnsoc:hasRenarrationTransformation
    [
      a rnsoc:RenarrationTransformation;
      oa:hasSelector
        [
          a oa:XPathSelector;
          rdf:value "/html[1]/body[1]/p[3]"
        ];
      rnsoc:createdAt "2020-12-05T11:03:28.005Z"^^xsd:dateTime;
      rnsoc:hasAction "replace"^^xsd:string;
      rnsoc:hasNarration
        [
          a rnsoc:Text;
          dc:format "text/html";
          dc:language "en";
          rnsoc:hasProvenance ex:external_document;
          oa:hasSelector
            [
              a oa:XPathSelector;
              rdf:value "/html[1]/body[1]/p[3]"
            ];
        ]
    ];
  rnsoc:onSourceDocument ex:source_document;
  rnsoc:renarratedBy c:me.
ex:source_document a rnsoc:Document.
```

Figure 7.34. A renarration specification with a single transformation that references a paragraph on an external document

rnsoc:Narration that is used for referencing a part of an external document operation can be seen between the Line:25 and the Line:36 in the Figure 7.34. The *rnsoc: hasProvenance* (Line:30) object property shows that the provenance of the narration content is the URI of external document in the Figure 7.33. The XPath selection value for the chosen `"<p>External Third Paragraph</p>"` element in the external document (Figure 7.33) is indicated with the *rdf:value* in the Line:34.

In the Figure 7.35 renarrated document can be observed. The change in the source document after the renarration process is in the Line:6 of the renarrated document.

```
1 <html>
2   <head></head>
3   <body>
4   <p>First Text</p>
5   <p>Second Text</p>
6   <p>External Third Paragraph</p>
7   </body>
8 </html>
```

Figure 7.35. A renarrated document obtained by applying the renarration specification shown in Figure 7.34 on the source document shown in Figure 7.2

7.2. Evaluation of the Model Using the Implemented Prototype

7.2.1. Use Case

This use case intends to demonstrate that the model and prototype are capable of creating renarration for a web document, and it enables social interactions between renarrators through responses. A sample stray dogs web page has been created to exemplify a translation and elaboration scenario, the web page can be seen in the Figure 7.37. The fictional scenario is about the sample web page of stray dogs, and it tells the story of a renarrator named Mete who would like to translate and elaborate the news for foreigners. The fictional news narrates the fictitious Foobar Municipality and provision of shelters for stray dogs. In the Figure 7.36 HTML document of stray dogs example, namely the source document can be seen.

```
 1 <html>
 2 <head>
 3   <meta http-equiv="Content-Type" content="text/html; charset=UTF
     -8">
 4   <link rel="stylesheet" href="stray_dogs_example.css">
 5 </head>
 6 <body>
 7   <div class="main">
 8     <h1>Foobar Belediyesi Sokak Köpeklerine Barınak Sağladı</h1>
 9     <div class="align-center">
10       <img src="straydogs.jpg" class="img-main">
11       <p class="caption">Sokak köpeklerinin yaşam koşulları</p>
12     </div>
13     <p class="p-text">Foobar Belediyesi yardımsever vatandaşlar tarafından
14       toplanan bağışlar sayesinde, sokak köpeklerine daha iyi
15       yaşayabilecekleri yeni yuvalar temin etti.Özenle hazırlanan bu
16       yuvaların içerisinde otomatik mama kapları dahi bulunmakta.
17       </p>
18   </div>
19 </body>
20 </html>
```

Figure 7.36. The source document of stray dogs example

Figure 7.37. The sample web page of stray dogs

Mete sees the news about Foobar Municipality and its provision of shelters for stray dogs on a news portal. He decides to renarrate this news for foreigners because some foreign friends don't understand his speak upon stray dogs. Since the local authorities catch and put them in shelters, there are no stray dogs in their country.

Mete starts the RNEx extension and logs into it with his Solid account using his WebID and password. He creates a renarration that contains four renarration transformations. Firstly, he clicks on the "Foobar Belediyesi Sokak Köpeklerine Barınak Sağladı" part and changes it to "Foobar Municipality Provided Shelters For The Stray Dogs" and creates the first renarration transformation with the "replace" action and "English" language. It can be seen between the Line:16 and the Line:33 of renarration in the Figure 7.39. Secondly, he translates the "Sokak köpeklerinin yaşam koşulları" to "Living conditions of the stray dogs" with the "replace" action and "English" language. It can be seen between the Line:34 and the Line:51.

The rest of the renarration specification in Figure 7.39 is shown in Figure 7.40. Thirdly, he clicks on the part which inscribes "Foobar Belediyesi yardımsever vatandaşlar tarafından toplanan bağışlar sayesinde, sokak köpeklerine daha iyi yaşayabilecekleri yeni yuvalar temin etti. Özenle hazırlanan bu yuvaların içerisinde otomatik mama kapları dahi bulunmakta.", and translates and puts additional information of location to it "Foobar Municipality, a municipality located in Istanbul, provided new shelters with automatic dog feeders for the stray dogs, with the help of donations from philanthropists." with the "replace" action and "English" language. It can be observed between the (Line:1) and the (Line:18). Fourthly, he adds the "Stray dogs can be seen commonly in various places in Turkey as well as stray cats." extra information with the "insert" action and "English" language. It can be observed between the (Line:19) and the (Line:46). Lastly, he chooses "translation" and "elaboration" options as a motivation of the renarration and saves his renarration. Stray dogs web page that is renarrated by Mete can be seen in the Figure 7.38. In the Figure 7.42 HTML document of stray dogs example after the appliance of renarration in Figure 7.39, namely the renarrated document, can be seen.

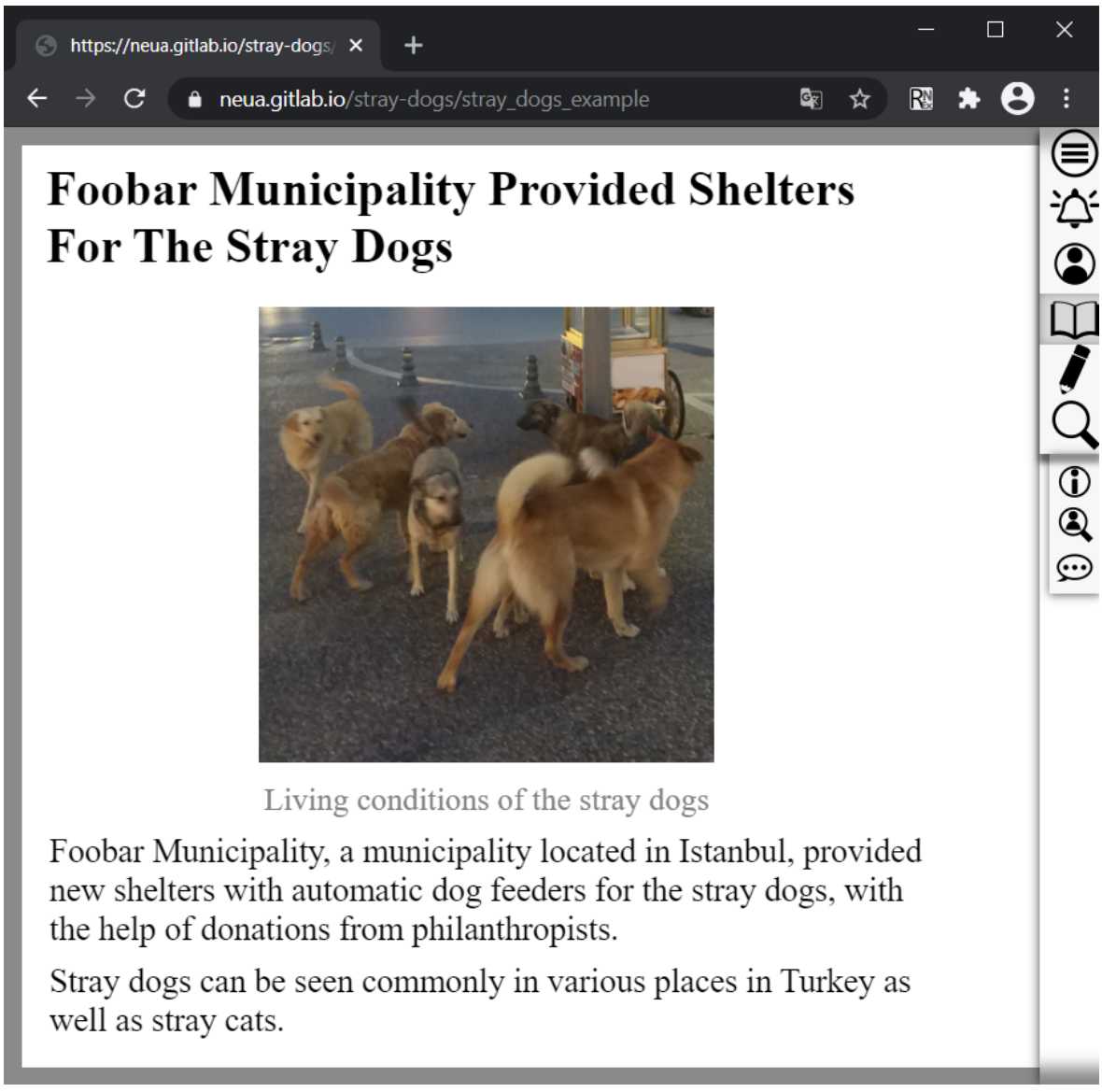

Figure 7.38. Stray dogs web page after the renarration process

```
@prefix : <#>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix owl: <http://www.w3.org/2002/07/owl#>.
@prefix rnsoc: <http://soslab.cmpe.boun.edu.tr/ontologies/rnsoc.owl>.
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.
@prefix oa: <http://www.w3.org/ns/oa#>.
@prefix dc: <http://purl.org/dc/elements/1.1/>.
@prefix str: <https://neua.gitlab.io/stray-dogs/>.
@prefix c: <https://mete-therenarrator.inrupt.net/profile/card#>.

:kjspe88p-w6vvqoon-bcuwk73q
    a owl:NamedIndividual, rnsoc:Renarration;
    rnsoc:createdAt "2021-01-11T15:10:53.497Z"^^xsd:dateTime;
    rnsoc:hasMotivation "elaboration"^^xsd:string, "translation"^^xsd:string;
    rnsoc:hasRenarrationTransformation
     [
      a rnsoc:RenarrationTransformation;
      oa:hasSelector
        [
          a oa:XPathSelector;
          rdf:value "/html[1]/body[1]/div[1]/h1[1]"
        ];
      rnsoc:createdAt "2021-01-11T15:10:05.931Z"^^xsd:dateTime;
      rnsoc:hasAction "replace"^^xsd:string;
      rnsoc:hasNarration
        [
          a rnsoc:;
          dc:format "text/html";
          dc:language "en";
          rdf:value
              "<h1>Foobar Municipality Provided Shelters For The Stray Dogs</h1>"
        ]
     ],
     [
      a rnsoc:RenarrationTransformation;
      oa:hasSelector
        [
          a oa:XPathSelector;
          rdf:value "/html[1]/body[1]/div[1]/div[1]/p[1]"
        ];
      rnsoc:createdAt "2021-01-11T15:10:12.241Z"^^xsd:dateTime;
      rnsoc:hasAction "replace"^^xsd:string;
      rnsoc:hasNarration
        [
          a rnsoc:;
          dc:format "text/html";
          dc:language "en";
          rdf:value
              "<p class="caption">Living conditions of the stray dogs</p>"
        ]
     ],

```

Figure 7.39. Renarration specification for the stray dogs web page first part

```
1       [
2        a rnsoc:RenarrationTransformation;
3        oa:hasSelector
4          [
5            a oa:XPathSelector;
6            rdf:value "/html[1]/body[1]/div[1]/p[1]"
7          ];
8        rnsoc:createdAt "2021-01-11T15:10:23.721Z"^^xsd:dateTime;
9        rnsoc:hasAction "replace"^^xsd:string;
10       rnsoc:hasNarration
11         [
12           a rnsoc:;
13           dc:format "text/html";
14           dc:language "en";
15           rdf:value
16               "<p class="p-text">Foobar Municipality, a
       municipality located in Istanbul, provided new shelters with
       automatic dog feeders for the stray dogs, with the help of
       donations from philanthropists.</p>"
17         ]
18       ],
19       [
20        a rnsoc:RenarrationTransformation;
21        oa:hasSelector
22          [
23            a rnsoc:RangeSelector;
24            oa:hasStartSelector
25              [
26                a oa:XPathSelector;
27                rdf:value
28                "/html[1]/body[1]/div[1]/p[1]"
29              ];
30            oa:hasEndSelector
31              [
32                a oa:XPathSelector;
33                rdf:value ""
34              ]
35          ];
36        rnsoc:createdAt "2021-01-11T15:10:43.423Z"^^xsd:dateTime;
37        rnsoc:hasAction "insert"^^xsd:string;
38        rnsoc:hasNarration
39          [
40            a rnsoc:;
41            dc:format "text/html";
42            dc:language "en";
43            rdf:value
44                "<p class="p-text">Stray dogs can be seen commonly
       in various places in Turkey as well as stray cats.</p>"
45          ]
46       ];
47      rnsoc:onSourceDocument str:stray_dogs_example;
48      rnsoc:renarratedBy c:me.
49 str:stray_dogs_example a rnsoc:Document.
```

Figure 7.40. Renarration specification for the stray dogs web page second part

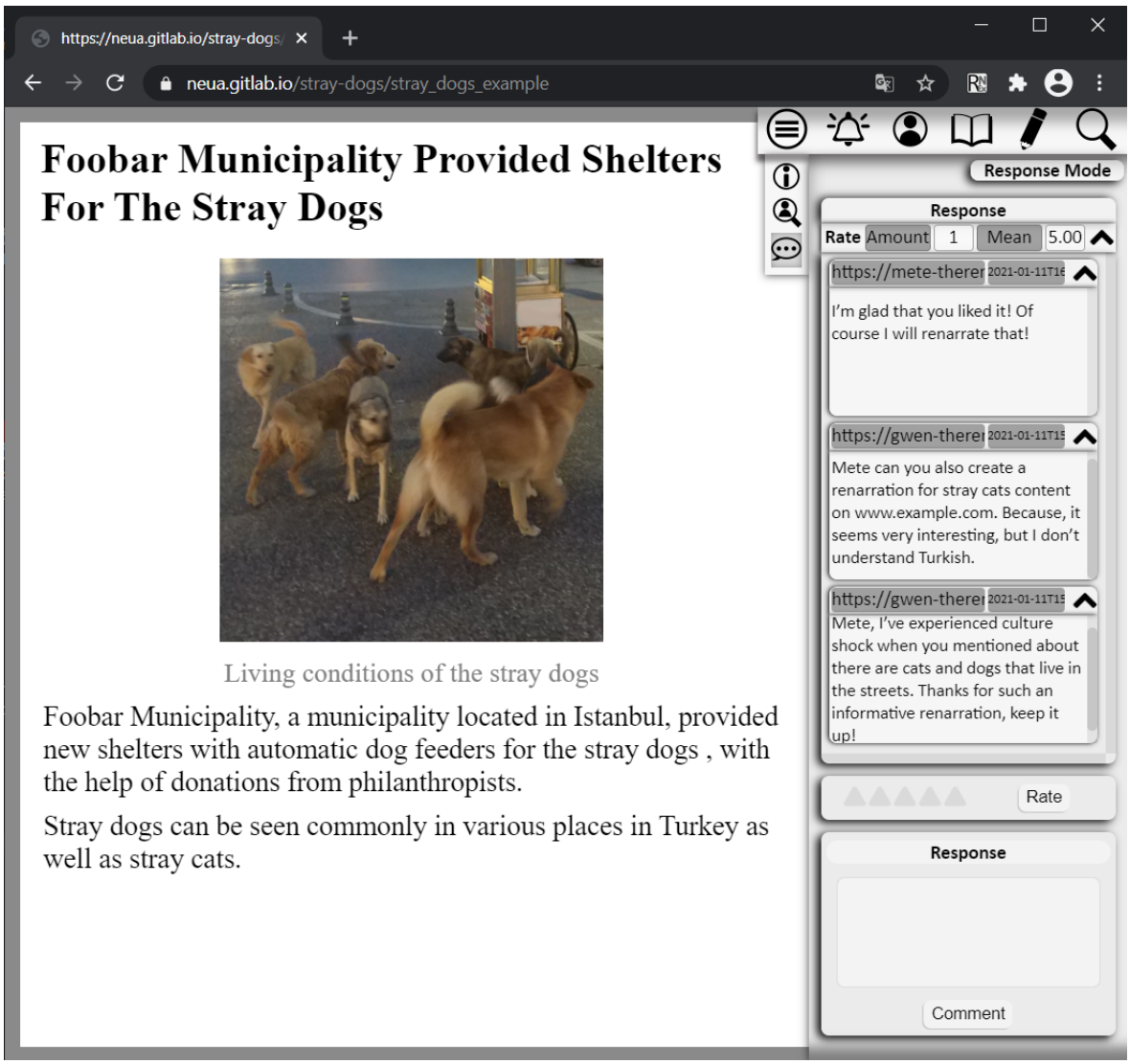

Figure 7.41. Response UI of RNEx for the stray dogs web page

```
 1 <html>
 2 <head>
 3   <meta http-equiv="Content-Type" content="text/html; charset=UTF
     -8">
 4   <link rel="stylesheet" href="stray_dogs_example.css">
 5 </head>
 6 <body>
 7   <div class="main">
 8     <h1>Foobar Municipality Provided Shelters For The Stray Dogs</
     h1>
 9     <div class="align-center">
10       <img src="straydogs.jpg" class="img-main">
11       <p class="caption">Living conditions of the stray dogs</p>
12     </div>
13     <p class="p-text">Foobar Municipality, a municipality located
     in Istanbul, provided new shelters with automatic dog feeders
     for the stray dogs, with the help of donations from
     philanthropists. Stray dogs can be seen commonly in various
     places in Turkey as well as stray cats.</p>
14     <p class="p-text">Stray dogs can be seen commonly in various
     places in Turkey as well as stray cats.</p>
15   </div>
16 </body>
17 </html>
```

Figure 7.42. The renarrated document obtained by applying the renarration specification shown in Figure 7.39 on the source document shown in Figure 7.36

After some time, another renarrator named Gwen sees the link of the Turkish news about the Foobar Municipality providing shelters for stray dogs. She becomes curious about the content but since she doesn't understand Turkish, looks for a renarration of it. Then sees that renarrator named Mete has a renarration about it then she clicks to read it.

She likes the renarration a lot because she gets informed about stray animals and culture in Turkey and understands the content that is written in another language. Since she likes the renarration a lot, she rates Mete's renarration 5 out of 5 and writes a comment "Mete, I've experienced culture shock when you mentioned about there are cats and dogs that live in the streets. Thanks for such an informative renarration, keep it up!". Then, she writes another comment "Mete can you also create a renarration for stray cats content on www.example.com. Because, it seems very interesting, but I don't understand Turkish." since she can't access the information in stray cats content in Turkish someone who knows the language should renarrate it, and she requests help from Mete to understand it.

The rate between the Line:12 and the Line:16, and the first comment between the Line:19 and the Line:24, and the second comment between the Line:27 and the Line:32, can be seen in turtle file in the Figure 7.43. RNEx response UI for the stray dogs web page and applied renarration can be seen in the Figure 7.41.

Subsequently, seeing the comment of Gwen, Mete feels contented to get a response which praises his renarration. Then he replies Gwen "I'm glad that you liked it! Of course I will renarrate that!" since Mete knows the culture and language she can help her access the information in the stray cats content. Mete's comment can be seen between the Line:14 and the Line:18 of renarration in the Figure 7.44.

```
@prefix : <#>.
@prefix c: <https://gwen-therenarrator.inrupt.net/profile/card#>.
@prefix foaf: <http://xmlns.com/foaf/0.1/>.
@prefix rnsoc: <http://soslab.cmpe.boun.edu.tr/ontologies/rnsoc.owl>.
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.
@prefix RNC: <https://mete-therenarrator.inrupt.net/renarration/
    RenarrationFiles/RNCNT-kjspe88p-w6vvqoon-bcuwk73q.ttl#>.
@prefix owl: <http://www.w3.org/2002/07/owl#>.

c:me a foaf:Person, rnsoc:Renarrator.

:kjspjb1i-t1chc6v4-6qu11ain
  a rnsoc:Rate;
  rnsoc:hasRateValue "5"^^xsd:int;
  rnsoc:isCreatedBy c:me;
  rnsoc:respondedAt "2021-01-11T15:14:50.406Z"^^xsd:dateTime;
  rnsoc:responseOnRenarration RNC:kjspe88p-w6vvqoon-bcuwk73q.

:kjspn0uv-7stb1i7z-80th00tr
  a rnsoc:Comment;
  rnsoc:hasCommentValue
    "Mete, I've experienced culture shock when you mentioned about
     there are cats and dogs that live in the streets. Thanks for
    such an informative renarration, keep it up!"^^xsd:string;
  rnsoc:isCreatedBy c:me;
  rnsoc:respondedAt "2021-01-11T15:17:43.831Z"^^xsd:dateTime;
  rnsoc:responseOnRenarration RNC:kjspe88p-w6vvqoon-bcuwk73q.

:kjsqefd4-qp5ivvuj-hies2zl1
  a rnsoc:Comment;
  rnsoc:hasCommentValue
    "Mete can you also create a renarration for stray cats content
     on www.example.com. Because, it seems very interesting, but I
     don't understand Turkish."^^xsd:string;
  rnsoc:isCreatedBy c:me;
  rnsoc:respondedAt "2021-01-11T15:39:02.344Z"^^xsd:dateTime;
  rnsoc:responseOnRenarration RNC:kjspe88p-w6vvqoon-bcuwk73q.

RNC:kjspe88p-w6vvqoon-bcuwk73q a owl:NamedIndividual, rnsoc:
    Renarration.
```

Figure 7.43. Gwen's response to the renarration of stray dogs web page

```
1 @prefix : <#>.
2 @prefix c: <https://mete-therenarrator.inrupt.net/profile/card#>.
3 @prefix foaf: <http://xmlns.com/foaf/0.1/>.
4 @prefix rnsoc: <http://soslab.cmpe.boun.edu.tr/ontologies/rnsoc.owl>.
5 @prefix RNC: <https://mete-therenarrator.inrupt.net/renarration//
    RenarrationFiles/RNCNT-kjkiaukl-p0l728xt-u4qzpyzs.ttl#>.
6 @prefix owl: <http://www.w3.org/2002/07/owl#>.
7 @prefix xsd: <http://www.w3.org/2001/XMLSchema#>.
8
9 c:me a foaf:Person, rnsoc:Renarrator.
10
11 RNC:kjkiaukl-p0l728xt-u4qzpyzs a owl:NamedIndividual, rnsoc:
    Renarration.
12
13 :kjkjnzfh-ac4jl0aq-8e5v5uqg
14   a rnsoc:Comment;
15   rnsoc:hasCommentValue "I'm glad that you liked it! Of course I
     will renarrate that!"^^xsd:string;
16   rnsoc:isCreatedBy c:me;
17   rnsoc:respondedAt "2021-01-11T16:17:11.521Z"^^xsd:dateTime;
18   rnsoc:responseOnRenarration RNC:kjkiaukl-p0l728xt-u4qzpyzs.
```

Figure 7.44. Mete's response to the renarration of stray dogs web page

7.3. Results and Discussion

The test cases examine the expressiveness of the Renarration Social Ontology in representing the desired renarrations and their relations to renarrators. The core specification of a renarration is expressed in the renarrations transformations, which define the alternative narrations. We have demonstrated how various transformations can be expressed with our ontology. We further shown how various social interactions can be represented. Use case evaluates the usage of Renarration Social Ontology in the prototype, RNEx, that enables creating renarrations and social interactions, with a renarration scenario that highlights the language and cultural barriers. The scenario also indicates that the use of social and semantic renarrations support semantic accessibility of content.

Accordingly, the Renarration Social Ontology creates a common ground for systems to use it as a base to manage social semantic renarration data. The use of ontology for representing the renarrations and social interactions enables semantic systems to reason over the social semantic renarration data and interoperability with other ontologies, and knowledge reuse [72]. Therefore, use of an ontological approach for representing social renarration platform domain facilitates connection to linked data datasets and obtaining information from different sources e.g., graph databases, knowledge graphs, triplestores.

The evaluations and proof of concept prototype RNEx show that Renarration Social Ontology is feasible for representing the creation of renarrations and social interactions of the renarration community and viable to use in the domain of social renarration platform that supports the semantic accessibility of content.

8. CONCLUSIONS AND FUTURE WORK

The vision of the Web was to facilitate the access to information to everyone across the globe [1]. The realization of this vision calls for the development of technologies and policies that support access to information by overcoming barriers. In this thesis, we focused on an approach to overcome semantic barriers on the web with an ontology driven model that supports the specification of renarrations of web content and interactions among the renarration community. For this purpose we proposed a social renarration platform that supports specifying alternative narrations that are machine-processable and interacting with renarration community. We benefited from several W3C recommended standards that address various needs of web content (mainly the Renarration Data Model [10], Web Annotation Data Model [13], and Activity Streams 2.0 [14]). We are encouraged by the presence of web standards and protocols that focus on the semantic representation of the content and how web communities interact with content. We are further encouraged by the ongoing decentralization and privacy protection protocols and projects like Solid [63]. Our work is the scope of fixing the Internet as is advocated by Tim Berners-Lee. More specifically, it aims to respond to the principle "by ensuring systematically excluded populations have effective paths towards meaningful internet access" articulated in the *contract for the web* [73].

We demonstrated the feasibility of our approach with a simple prototype which is implemented as a browser extension. Browser extensions are suitable for this purpose as they can address the experience of individual users, such as customized renarration recommendations. To address the decentralization and privacy issues we used Solid, which focuses on issues of great significance to web users. Solid supports RDF representations, which makes it naturally very appealing for our purposes. Unfortunately, the Solid project is not as mature as we would have hoped. The main shortcoming for our purposes was the lack of full SPARQL support for searching across several pods. This is important in our case since renarration specifications are distributed across the pods of renarrators. We had to devise some strategies to overcome this problem. Solid has provided a good starting point for delivering a decentralized system that stores

data in a semantically structured manner that also provides mechanisms for owner-controlled access rights and true data ownership. Despite of its current shortcomings, we are satisfied with using Solid for the benefits it offers. The developments on Solid are ongoing and we expect this and other issues to be resolved.

One concern is the potential abuse of the social renarration platform to create renarrations with the intent to fool others. We have taken precautions that explicitly represent the original source, the renarrator, renarration date, and the specifications of transformations that work only when the original source is accessible. The transformations do not capture content from the source document, but rather, specify them with an addressing scheme that identifies a fragment within a source document. Renarration specifications maintains the provenance of source documents and are useless when source documents are not available (via removal or change). Thus, without access to the original source document, a renarrated document can not be rendered. This will not prevent an ill-intended user to save and publish a renarrated document. Such abusive use is a persistent concern regarding web content. Solving this issue is beyond the scope of this work.

There remain several research directions that can be summarized as follows:

- RNEx and Renarration Social Ontology can be extended to support:
  - renarration specifications that are defined on multiple source documents,
  - collaborative renarration, and
  - renarrations of renarration
- The renarration transformations could be used to detect patterns that could be used as automated rewriting rules. This way, the wisdom of renarrators could be utilized to provide some automated renarrations, such as in recipes, common descriptions, and cultural customs.
- Issues related to spamming, cyberbullying, trust, security, relatedness of renarrations can be addressed by developing trust models based on the quality of the renarrations and social interactions.
- The searching and browsing functionality of the social renarration platform should

support more sophisticated semantic processing. As the Solid pod's support for semantic queries is limited, we could not perform various queries. Therefore, conducting sophisticated federated queries across pods would be a significant development.

- A recommendation model that utilizes the social network analysis of the renarrations and renarrators, as well as the renarration transformations, should be developed. Users should be able to get recommendations for renarrations based on their interests and profiles (such as language, education).
- The social renarration platform could be extended to offer useful suggestions to renarrators while they specify renarrations. Such suggestions may include definitions, synonyms, semantic properties obtained from Wikidata [74], DBPedia [75].
- Our proof of concept prototype (RNEx) supports renarrations at the HTML element level. Clearly, more fine-grained selections are required to provide high-quality renarrations.
  - User should be able to make selections at the desired level of resolution.
  - When a renarration transformation related to a fragment within the element is applied they must preserve the context of that element. For example, if a word within a paragraph element is replaced with another, the resulting renarration should appear exactly the same for the source document except for the replaced word (i.e all style and content).

In summary, this work introduced a social semantic web approach for extending web accessibility. We believe that capturing information semantically offers many interesting opportunities for collaboration, recommendations, and collective intelligence. Renarrations in the context of the ever expanding Linked Data [70] offer exciting possibilities.

REFERENCES

1. WAI, W., *Introduction to Web Accessibility*, https://www.w3.org/WAI/fundamentals/accessibility-intro/, accessed in January 2021.

2. Miniwatts Marketing Group, *Internet Growth Statistics*, https://www.internetworldstats.com/emarketing.htm, accessed in January 2021.

3. W3C, *W3C MISSION*, https://www.w3.org/Consortium/mission, accessed in August 2020.

4. Wikipedia contributors, *Web 2.0 — Wikipedia, The Free Encyclopedia*, 2021, https://en.wikipedia.org/w/index.php?title=Web_2.0&oldid=1000364874, accessed in January 2021.

5. CIAT, *"We are not alone": Nicaragua's rural youth tell their story*, https://blog.ciat.cgiar.org/we-are-not-alone-nicaraguas-rural-youth-tell-their-story/, accessed in January 2021.

6. Kanan Dhru, *First Information Report*, https://prezi.com/fxk9h96h0zyp/first-information-report/, accessed in January 2021.

7. Khan Academy, *Fourier Series introduction*, https://youtu.be/UKHBWzoOKsY, accessed in January 2021.

8. Dinesh, T., S. Uskudarli, S. Sastry, D. Aggarwal and V. Choppella, "Alipi: A framework for re-narrating web pages", *Proceedings of the International Cross-Disciplinary Conference on Web Accessibility*, pp. 1–4, 2012.

9. Prasad, G. V. S., T. Dinesh and V. Choppella, "Overcoming the new accessibility challenges using the sweet framework", *Proceedings of the 11th Web for All*

Conference, pp. 1–4, 2014.

10. Guder, E., *Increasing Accessibility of Web Content via Semantic Renarration*, Master's Thesis, Bogazici University, 2016.

11. Guder, E. and S. Uskudarlı, *Renarration Data Model*, `http://soslab.cmpe.boun.edu.tr/ontologies/rn.owl`, accessed in January 2021.

12. Hos, E. and S. Uskudarlı, *Renarration Social Ontology*, `http://soslab.cmpe.boun.edu.tr/ontologies/rnsoc.owl`, accessed in January 2021.

13. W3C, *Web Annotation Data Model*, `https://www.w3.org/TR/annotation-model/`, accessed in August 2020.

14. W3C, *Activity Streams 2.0*, `https://www.w3.org/TR/activitystreams-core/`, accessed in August 2020.

15. Wikipedia contributors, *Browser extension — Wikipedia, The Free Encyclopedia*, 2020, `https://en.wikipedia.org/w/index.php?title=Browser_extension&oldid=989500301`, accessed in December 2020.

16. Hos, E., *RNEx - A decentralized social renarration platform*, `https://gitlab.com/neua/RNEx`, accessed in January 2021.

17. Sambra, A. V., E. Mansour, S. Hawke, M. Zereba, N. Greco, A. Ghanem, D. Zagidulin, A. Aboulnaga and T. Berners-Lee, *Solid: a platform for decentralized social applications based on linked data*, Tech. rep., Technical report, MIT CSAIL & Qatar Computing Research Institute, 2016.

18. W3C, *RDF*, `https://www.w3.org/RDF/`, accessed in April 2020.

19. Wikipedia contributors, *Solid (web decentralization project) — Wikipedia, The Free Encyclopedia*, 2020, `https://en.wikipedia.org/w/index.php?title=Solid_(web_decentralization_project)&oldid=942361189`, accessed in March 2020.

20. Solid, *Web Access Control (WAC)*, https://github.com/solid/web-access-control-spec/, accessed in October 2020.

21. Wikipedia contributors, *Ontology (information science) — Wikipedia, The Free Encyclopedia*, 2020, https://en.wikipedia.org/w/index.php?title=Ontology_(information_science)&oldid=943552529, accessed in April 2020.

22. W3C, *Vocabularies*, https://www.w3.org/standards/semanticweb/ontology.html, accessed in April 2020.

23. Wikipedia contributors, *Resource Description Framework — Wikipedia, The Free Encyclopedia*, 2020, https://en.wikipedia.org/w/index.php?title=Resource_Description_Framework&oldid=946199997, accessed in April 2020.

24. W3C, *RDF Schema 1.1*, https://www.w3.org/TR/rdf-schema/, accessed in April 2020.

25. Wikipedia contributors, *RDF Schema — Wikipedia, The Free Encyclopedia*, 2019, https://en.wikipedia.org/w/index.php?title=RDF_Schema&oldid=929101599, accessed in April 2020.

26. W3C, *OWL*, https://www.w3.org/OWL/, accessed in April 2020.

27. Wikipedia contributors, *Web Ontology Language — Wikipedia, The Free Encyclopedia*, 2020, https://en.wikipedia.org/w/index.php?title=Web_Ontology_Language&oldid=946233896, accessed in April 2020.

28. Wikipedia contributors, *SPARQL — Wikipedia, The Free Encyclopedia*, 2020, https://en.wikipedia.org/w/index.php?title=SPARQL&oldid=949015970, accessed in April 2020.

29. W3C, *SPARQL Query Language for RDF*, https://www.w3.org/TR/rdf-sparql-query/, accessed in April 2020.

30. Ontotext, *What is SPARQL?*, https://www.ontotext.com/knowledgehub/fundamentals/what-is-sparql/, accessed in April 2020.

31. Wikipedia contributors, *Turtle (syntax) — Wikipedia, The Free Encyclopedia*, 2020, https://en.wikipedia.org/w/index.php?title=Turtle_(syntax)&oldid=944970780, accessed in April 2020.

32. W3C, *RDF 1.1 Turtle*, https://www.w3.org/TR/turtle/, accessed in March 2020.

33. Google, *What are extensions?*, https://developer.chrome.com/extensions, accessed in July 2020.

34. Google, *Getting Started Tutorial*, https://developer.chrome.com/extensions/getstarted, accessed in July 2020.

35. WHATWG, *HTML Living Standard*, https://html.spec.whatwg.org/, accessed in December 2020.

36. Group, C. W., *Cascading Style Sheets home page*, https://www.w3.org/Style/CSS/, accessed in December 2020.

37. Mozilla and individual contributors, *JavaScript*, https://developer.mozilla.org/en-US/docs/Web/JavaScript, accessed in December 2020.

38. MIT and other contributors, *Rdflib*, https://linkeddata.github.io/rdflib.js/doc/, accessed in April 2020.

39. MIT and other contributors, *What is rdflib*, https://linkeddata.github.io/rdflib.js/Documentation/webapp-intro.html, accessed in April 2020.

40. Solid, *A library for reading and writing to Solid pods*, https://github.com/solid/solid-auth-client/, accessed in October 2020.

41. W3C, *WebID*, https://www.w3.org/wiki/WebID, accessed in October 2020.

42. Wikipedia contributors, *WebID — Wikipedia, The Free Encyclopedia*, 2020, https://en.wikipedia.org/w/index.php?title=WebID&oldid=964951846, accessed in October 2020.

43. Dan Brickley, L. M., *FOAF Vocabulary Specification 0.99*, http://xmlns.com/foaf/spec/, accessed in December 2020.

44. Wikipedia contributors, *FOAF (ontology) — Wikipedia, The Free Encyclopedia*, 2020, https://en.wikipedia.org/w/index.php?title=FOAF_(ontology)&oldid=988308362, accessed in December 2020.

45. Dublin Core™ Metadata Initiative, *DCMI Metadata Terms*, https://www.dublincore.org/specifications/dublin-core/dcmi-terms/, accessed in December 2020.

46. Wikipedia contributors, "Dublin Core — Wikipedia, The Free Encyclopedia", , 2020, https://en.wikipedia.org/w/index.php?title=Dublin_Core&oldid=996438175, accessed in December 2020.

47. W3C, *What is an XML Schema?*, https://www.w3.org/standards/xml/schema, accessed in December 2020.

48. Wikipedia contributors, "XML Schema (W3C) — Wikipedia, The Free Encyclopedia", , 2020, https://en.wikipedia.org/w/index.php?title=XML_Schema_(W3C)&oldid=961489334, accessed in December 2020.

49. Dinesh, T. and S. Uskudarli, "Renarration for All", *arXiv preprint arXiv:1810.12379*, 2018.

50. WHATWG, *DOM Living Standard*, https://dom.spec.whatwg.org/, accessed in December 2020.

51. Dinesh, T., V. Choppella and S. Uskudarli, "Re-narration as a basis for accessibility and inclusion on the World Wide Web", `http://dinesh.servelots.com/alipi/AlipiWWW.pdf`, accessed in January 2021.

52. Dinesh, T. and S. Uskudarli, "A social web for another billion", *Proceedings of M4D 2012 28-29 February 2012 New Delhi, India*, Vol. 28, No. 29, p. 223, 2012.

53. Hypothesis, *To enable a conversation over the world's knowledge*, `https://web.hypothes.is/about/`, accessed in September 2020.

54. Hypothesis, *Hypothesis browser extension(s)*, `https://github.com/hypothesis/browser-extension`, accessed in September 2020.

55. Hypothesis, *Embedding Hypothesis in Websites and Platforms*, `https://web.hypothes.is/help/embedding-hypothesis-in-websites-and-platforms/`, accessed in January 2021.

56. W3C, *Web Annotation Working Group*, `https://www.w3.org/annotation/`, accessed in September 2020.

57. Genius, *Web-annotator*, `https://genius.com/web-annotator`, accessed in October 2020.

58. Annotator contributors, *Annotator*, `http://annotatorjs.org/`, accessed in October 2020.

59. Capadisli, S., A. Guy, R. Verborgh, C. Lange, S. Auer and T. Berners-Lee, "Decentralised authoring, annotations and notifications for a read-write web with dokieli", *International Conference on Web Engineering*, pp. 469–481, Springer, 2017.

60. Capadisli, S., *Dokieli*, `https://dokie.li/`, accessed in October 2020.

61. W3C, *Linked Data Platform 1.0 Primer*, `https://www.w3.org/TR/ldp-primer/`, accessed in October 2020.

62. W3C, *Linked Data Notifications*, https://www.w3.org/TR/ldn/, accessed in October 2020.

63. Solid, *What is Solid?*, https://solid.mit.edu/, accessed in October 2020.

64. W3C, *XML Path Language (XPath)*, https://www.w3.org/TR/1999/REC-xpath-19991116/, accessed in January 2021.

65. Noy, N. F., D. L. McGuinness *et al.*, *Ontology development 101: A guide to creating your first ontology*, 2001.

66. Pinto, H. S. and J. P. Martins, "A methodology for ontology integration", *Proceedings of the 1st international conference on Knowledge capture*, pp. 131–138, 2001.

67. W3C, *RDF 1.1 XML Syntax*, http://www.w3.org/TR/rdf-syntax-grammar/, accessed in December 2020.

68. W3C, *OWL 2 Web Ontology Language Primer (Second Edition)*, https://www.w3.org/TR/owl-primer/, accessed in December 2020.

69. W3C, *W3C XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes*, https://www.w3.org/TR/xmlschema/, accessed in December 2020.

70. Wikipedia contributors, *Linked data — Wikipedia, The Free Encyclopedia*, 2020, https://en.wikipedia.org/w/index.php?title=Linked_data&oldid=983882311, accessed in October 2020.

71. Zucker, J., "Solid-File-Client", https://github.com/jeff-zucker/solid-file-client/, accessed in October 2020.

72. Chandrasekaran, B., J. Josephson and V. R. Benjamins, "What Are Ontologies, and Why Do We Need Them?", *Intelligent Systems and their Applications, IEEE*, Vol. 14, pp. 20 – 26, 02 1999.

73. World Wide Web Foundation, *Contract for the Web*, `https://contractfortheweb.org/`, accessed in January 2021.

74. WikiData, *Welcome to Wikidata*, `https://www.wikidata.org/wiki/Wikidata:Main_Page`, accessed in January 2021.

75. DBpedia, *DBpedia*, `https://wiki.dbpedia.org/`, accessed in January 2021.