**Providing Annotation Functionality in Loadstar**

*Ahmet Melek – Bogazici University*

1. **Introduction**
   1. Introduction to Semantic Technologies with Loadstar Examples
   2. Annotations and Motivations on Providing Annotation Functionality
2. **The Model**
   1. Data Pods and Annotations
   2. The Model – Technical Details
      1. Pods – Read & Write Permissions
      2. Vocabulary
      3. Data Model
      4. Protocols
3. **The Method** 
   1. Annotation Types and Managing Social Dilemmas
   2. Interface & End User Perspective
4. **Social Characteristics**
   1. Accesibility
   2. Note Taking
   3. Detailization
   4. Referencing and Reviewing
5. **Semantic Characteristics**
6. **An Example Case**
7. **Further Discussion**
8. **Conclusion**
9. **References**
10. **Introduction**

1.1 A Brief Introduction on Semantic Web Technologies with Loadstar Platform Examples

**Loadstar** is a platform in which users can create **tables** about any kind of concept. [[1]](#footnote-1)

These **tables** could be formed to describe the **concept**, taking notes on a concept, making lists etc.

In the backend, table data is stored in a **Triplestore** database. A triplestore differs from a relational database (such as mySQL) in the sense that it stores **triples** instead of tables with rows. Loadstar visualizes the triple data in tables.

* In a **table**, the **subject** is constant,meaning, all **triples** contain the same subject. Subjects are also optional, if a subject is not provided by the user, it is a blank node.
* The **context** of each and every triple in the table is the table URI: loadstar.com/username/table\_name

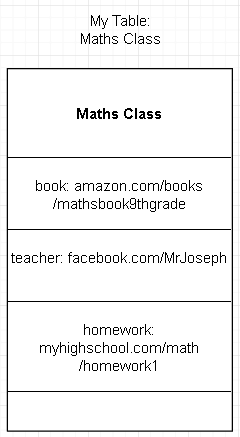


Figure 1: A table could be such as this.

(the subject is defined by user, however, not shown on the table.)

Every row in the table represents a triple.

In this case, the table is named `Maths Class`. In the table, there are three triples:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

s = dbpedia.org/resource/Mathematics

p = loadstar.com/ontology/book First Triple

o = amazon.com/books/mathsbook9thgrade

context = loadstar.com/ahmet/maths\_class

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

s = dbpedia.org/resource/Mathematics

p = loadstar.com/ontology/teacher Second Triple

o = facebook.com/MrJoseph

context = loadstar.com/ahmet/maths\_class

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

s = dbpedia.org/resource/Mathematics

p = loadstar.com/ontology/homework Third Triple

o = myhighschool.com/math/homework1

context = loadstar.com/ahmet/maths\_class

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

The subject of the table provides us with the knowledge of ‘Which topic is this table about?’. This is useful when making SPARQL queries, analyses etc.

loadstar.com/ontology is the root path of all ontology elements of Loadstar.

Loadstar also has moderators, which create encyclopedic tables to accumulate knowledge on the platform. With tools that Loadstar provides, data could be pulled automatically from various resources such as DBpedia or Wikidata, in order to form tables in Loadstar.

In the Loadstar platform, there are user profiles.

“Loadstar Platform” has a user profile of its own, located in [www.loadstar.com/users/Loadstar](http://www.loadstar.com/users/Loadstar) .

User profile of Loadstar contains the encyclopedic tables that are created by the moderators.

When a keyword is searched on Loadstar Platform, tables that are generated by the users will appear based on the popularity and the rating of the tables.

This way Loadstar provides an information serving ecosystem with both manually managed content, created by mods; and natural content created by users.

The mission of Loadstar Platform is to guide people on an endless journey of learning, with an environment that favors diversity of opinions.

The main feature of the platform which is designed to attract users in the initial stage is that, platform is an easy-to-use table forming and table finding tool.

1.2 Annotations and Motivations on Providing Annotation Functionality

Annotation Recommendations are standards that are provided by World Wide Web Consortium. They provide the standards to develop annotation functionality, that is, the features that enable annotating a part of a document with text, images, videos or parts of other documents.

An annotation is a type of object that bridges a target and a body. The target is the selected part of the main document (selected to be annotated), and the body is the note that is left on the target. Both target and body is to be represented with URIs.

Loadstar has a variety of reasons to implement this kind of feature, such as providing accessibility, improving user experience, providing extra functionality to enrich accumulated data, and to link accumulated data with Linked Data better.

* When `people` are considered to be provided with a learning ecosystem, accessibility plays a significant role. The masses that have a slow internet connection, people that are handicapped or people that have problems with understanding the lingua franca of the internet are not to be left behind. Annotation could provide further information for people having difficulties with consuming the existing information.
* Users are to have more freedom and opportunity of creativity with this extra feature. It would make Loadstar more of a beneficial environment.

To give an example, a user could make a table on his/her favorite books (subject of each and every triple in the table would be the user itself, predicates would be “loadstar:favoritebook”), select the books as annotation targets, and paste the authors’ names on the annotation bodies.

* Data created on Loadstar could get bigger in size and get more diverse with annotations.
* Annotations would be one of the rare sources to make sentiment analysis on Loadstar.
* Annotations are also a good way to link data. If the body of the annotation is a part of another document, the target document and the body document is being linked with an indirect relation, that is, being connected to the same annotation object. Figure 2 is shown below to illustrate this.

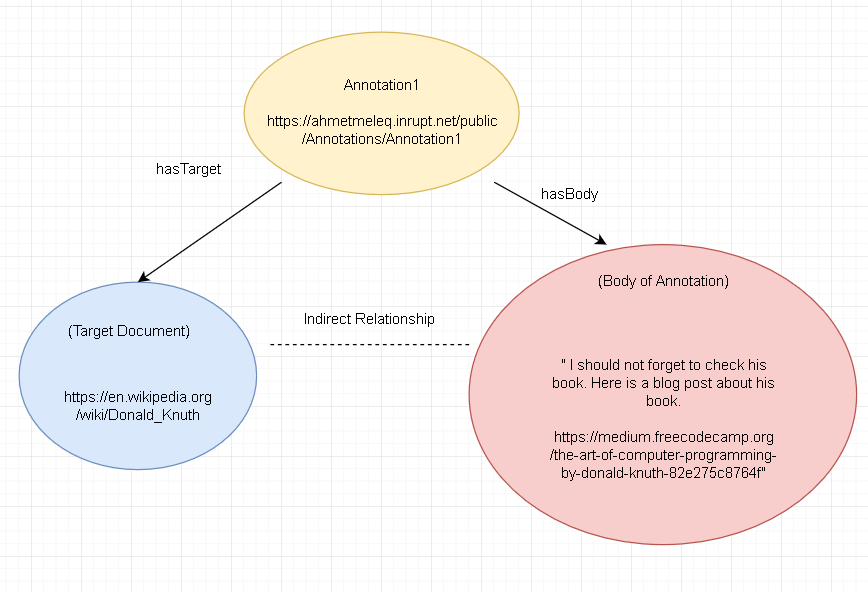


Figure 2: Modelling of the Annotation. [[2]](#footnote-2)

In Figure 2, a user puts an annotation onto the Wikipedia Article of Donald Knuth (annotation is on wikipedia, not loadstar). In this case, body of the annotation is a string containing a link.

When the user gets into the article of Donald Knuth in Wikipedia, given that a browser extension for annotating is installed, he/she sees the annotation on the Wikipedia article and remembers to check the book. Annotations could be used as reminders, such as in this case, however there are many other potential uses of annotations such as reviewing, taking notes, referencing etc. The Annotation protocols, vocabulary and data model provided by W3C are standards for annotation applications to follow.

Notice that in the above example (Figure 2), annotation object itself is hosted on a site named Inrupt. Inrupt and its potential benefits when used with annotations will be discussed in the next chapter.

1. **The Model**
   1. Data Pods and Annotations

SOLID is a web decentralization project led by Tim Berners Lee.[[3]](#footnote-3) It is an acronym for Social Linked Data. Solid provides Pods for users, to let users have a space to store their own data. When users register to SOLID web applications , the applications would ask for permission to access the data in user’s Pod. Inrupt is a startup that is founded to provide an ecosystem to help protect the integrity and quality of the new Web, built which is built on SOLID.[[4]](#footnote-4)

It is similar to the case when applications ask permissions to use data in users’ mobile phones. The applications do not store user’s pictures, notes, calendar info or contacts, but ask for user permission to use that data, stored in users’ phones.

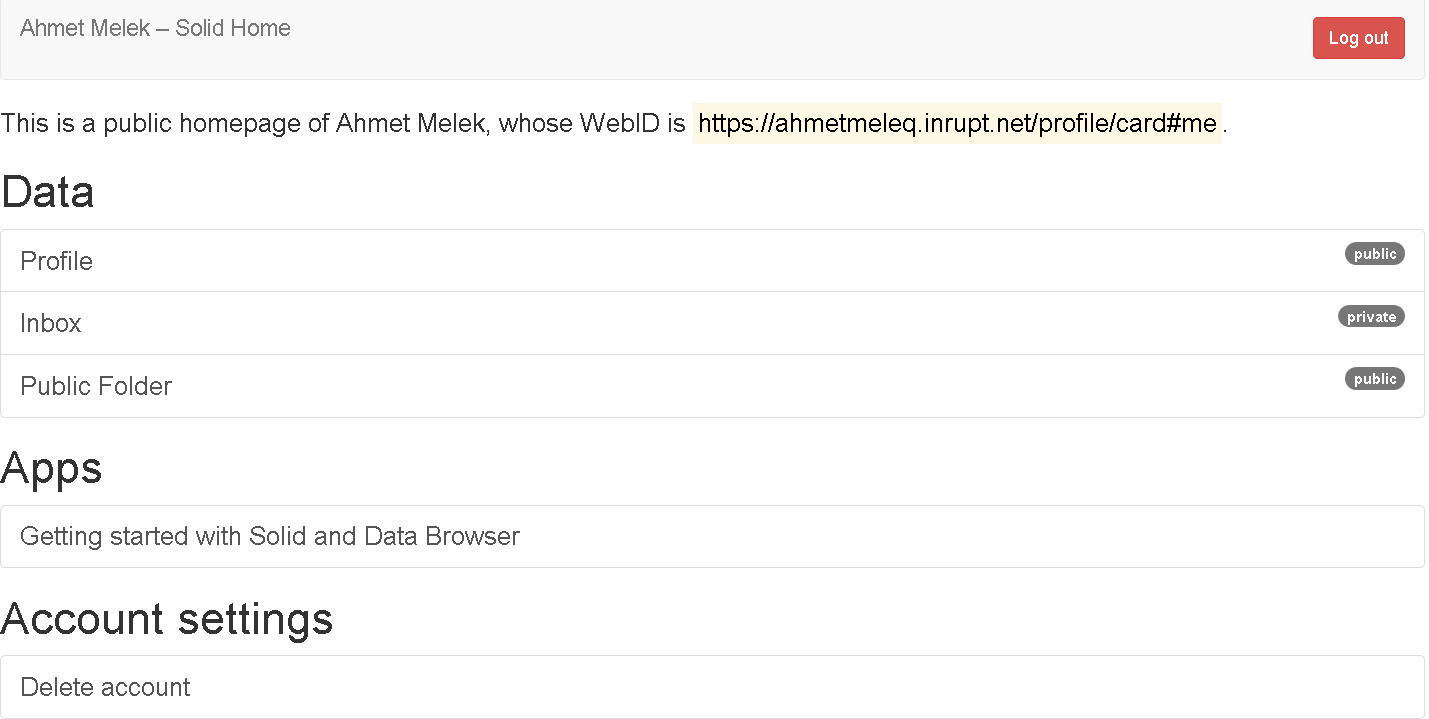
Pods have a lot of potential in terms of augmenting Linked Data and providing more control to users on their own data, thus augmenting security and privacy. However the main aspect of the Pods in our case is augmenting the data that is created in Loadstar, and doing this without making users store their annotation data in annotation applications’ (such as browser extensions) databases.

In Figure 2 of previous section, it can be seen that the annotation object itself is located in user’s Pod. In that case, Loadstar is not involved with the annotation, it is only that user makes an annotation on Wikipedia and stores the annotation object in the Pod.

In the case of Loadstar, platform will ask for permission to reach user’s pod (the section in pod where annotations are located) and platform will use the data in the Pod to show annotations on the Loadstar table. Implementing annotations this way instead of storing annotations in Loadstar databases has two advantages. Before diving into those reasons, let’s talk about Linked Data.

Linked Data is a web of data that is formed by the people who use semantic technologies, to form relationships between their data and data shared by other people. It includes more than 1200 databases and is growing[[5]](#footnote-5). It helps programmers to query data on the web with SPARQL language. It is analogous to a global brain in the sense that it includes many types of information from all around the globe, and that information is semantically linked.

Since pods use semantic technologies to represent the data, they are queryable by semantic query languages (such as SPARQL). Pods are much likely to contain a diverse collection of data, since the data accumulators are humans instead of machines. This diversity of data enables programmers to make more complex queries, using the pods as bridges in their queries. In a nutshell, **pods are bridges that are rich in connections and they are beneficial nodes for traversing Linked Data** with query languages.



Ahmet Melek’s pod on ahmetmeleq.inrupt.net

Second advantage that pods introduce is **user flexibility.** Users may want to aggregate their annotation data. Users may not want to allow Loadstar to store their annotation data, or may want to grant this permission for only a certain period of time. Power of users in such cases depend on the way the data is stored.

When the data is stored inside a certain application’s servers such as Loadstar’s, freedom for users that is mentioned above is not to be actualized fully. Applications could claim to support users’ freedom, however pods guarantee this by design. This decision should not be left to applications. The halter should be in users’ hands. Also, aggregating user data in users’ Pods could improve the data quality in the sense that all data is linked to one user profile. Since aggregated data is more valuable than fragmentary data, applications could utilize user data better, if they have permissions for a big number of users. Both applications and users are to benefit from the monolithic data aggregated in the pods.

2.2 The Model – Technical Details

Our model is best to be compliant with W3C’s standards, while transferring, parsing and storing annotation data. Otherwise the potential performance of annotations are not to be realized fully. These standards include a vocabulary to name classes and relationship types when annotation data is formed.

First, we wil present Pods which are key elements to the SOLID project. Then, Vocabulary is presented in order to familiarize the reader with the concepts to be encountered in the following sub-chapters. Those sub-chapters are the Data Model and the Protocols.

Vocabulary can be thought as ontology for annotations. W3C defines this vocabulary so there is a consensus on annotation terms. Significance of this situation lies in reaching Linked Data’s full potential (with having consensus on vocabulary). When everyone agrees on the vocabulary, there is no data that looks strange to anyone. Everyone is already familiar with the classes or relationship types. This provides Interoperatibility between those who are concerned with data.

Another standard of W3C is on the data model of annotations. This is again to provide consensus for cases when Linked Data is processed by unfamiliar parties. Vocabulary is about the terms, and data model is about using the vocabulary correctly to represent the data.

Last standard is on data transfer protocols. It can be described as which HTTP methods must be supported in which kinds of scenarios, and what kinds of responses should be returned when various requests are made.

# 2.2.1 Pods – Read&Write Permissions

To use user pods to store annotations, Loadstar asks for permission to Read&Write on the pod. It uses a data model reliant to W3C Annotation Data Model and writes onto Users’ Pod accordingly. Loadstar uses solid-auth-client library in order to perform reading and writing actions on the Pod.

Within every annotation URI; GET, HEAD and OPTIONS requests must be supported according to W3C’s documentation on Protocols.[[6]](#footnote-6) It is unknown if Inrupt profiles support HEAD and OPTIONS requests yet. (Please check Appendix1 at the end of the document.) For the sake of the project, it is assumed that Inrupt profiles do support HEAD and OPTIONS requests.

Solid-auth-client library is able to track sessions.

Library also provides a login-logout functionality, and fetch functionality to make requests on Solid Pods.[[7]](#footnote-7)

Fetch function can be used this way to write on a user pod[[8]](#footnote-8):

solid.auth.fetch(

'https:// ahmetmeleq.inrupt.net/public/Annotations/load\_anno.ttl',

{

method: 'PUT',

body: data

} )

.then(res => {return res;});

And used this way to read the data from a user pod:

solid.auth.fetch(

'https:// ahmetmeleq.inrupt.net/public/Annotations/load\_anno.ttl',

{

method: 'GET',

} )

.then(res => {return res;});

# 2.2.3 Vocabulary (Ontology)[[9]](#footnote-9)

Vocabulary the platform uses is clearly documented in W3C’s site. There are 9 classes and 13 properties Loadstar implements from this ontology. In this section, they are explained briefly in order to clarify them for the next chapters.

**Classes:**

2.1.1 Annotation: Main class for the annotation object

2.1.7 FragmentSelector: To selecta part of any type of Target. Loadstar mainly uses it for videos.

2.1.10 RangeSelector: To select a beginning and and end for the annotation. We use it with

Xpath selector together in order to make selections on DOM.

2.1.12 Selector: Main class for selectors.

2.1.16 SvgSelector: To select polynomial areas from SVG images.

2.1.18 TextQuoteSelector: To select texts with copying them.

2.1.19 TextualBody: To represent text bodies.

2.1.21 XpathSelector: Used with range selector to select items on DOM.

**Properties:**

2.2.2 bodyValue: Value is a plain text body.

2.2.5 end: Value is the end of the TextQuoteSelector class. In order to detect the

correct text.

2.2.6 exact: Value is the text of TextQuoteSelector class.

2.2.7 hasBody: Value is the body.

2.2.8 hasEndSelector: Value is the end of the RangeSelector class.

2.2.11 hasSelector: Value is the Selector class or a subclass of it.

2.2.12 hasSource: Value is the URI of the target when target node is blank and used as a

bridge between selector and the source.

2.2.13 hasStartSelector: Value is the beginning of the RangeSelector class.

2.2.15 hasTarget: Value is the target.

2.2.17 prefix: The beginning of the TextQuoteSelector class in order to detect the

correct text.

2.2.18 processingLanguage: To support different languages, values are strings indicating language.

2.2.27 suffix: The endıng of a selector.

2.2.29 via: Main source of the Annotation. Platform uses this in order to state the

Pods as the primary resource for the annotations .

# 2.2.2 Data Model[[10]](#footnote-10)

As it would increase congestion to include all the properties of all classes here, it is advised to check them on W3C’s documentation. The Data Model is explained within the 6th Chapter with the Example Case, to make concepts clearer. (Check 6th Chapter)

Some Notes:

* Data Model is usually serialized in JSON-LD format, as a standart of W3C Data Model.
* Main concept in the Data Model is the class “Annotation” which has a body and a target.
* Data Model offers such extra concepts:
  + Choice between several bodies (to use the anno with: ie. different languages)
  + Time States of a Document
  + Selectors (to select a particular region on any type of data)
  + Motivation Relation (to know about the intent of the user forming the annotation)
  + Styles (to render in a particular style)
  + Accesibility (to include strings that describe the accesibility of the document)
* According to Cardinality principle, if there is an Annotation consisting of multiple bodies and targets, each body is related to each target equally.

# 2.2.4 Protocols

As it would increase congestion to include all the request-repsonse types here, it is advised to check them on W3C’s documentation. The Protocols are explained within the 6th Chapter with the Example Case, to make use cases clearer. (Check 6th Chapter)

Some Notes:

* Issue of the protocols can be seperated into two:
  + Protocols for Reading Data
  + Protocols for Writing Data
    - For Creating
    - For Updating
    - For Deleting
* A service for managing annotations is called an Annotation Container and it is a must for an Annotation Server to have an Annotation Container.
* All containers must support GET, HEAD and OPTIONS methods as received requests .
* For the discovery of Annotation Containers, it is important for Web Server to return a link to an Annotation Container in case of the resource being linked to a Annotation Container. (As a response to every ordinary )
* Annotation Containers could include pages to segment data. If this is the case, the response to the request on container URL with page parameters is the page object and the items it contains. These items could be embedded inside the page or not. (In the latter case, the IRIs of the items are returned with the page.) [[11]](#footnote-11)

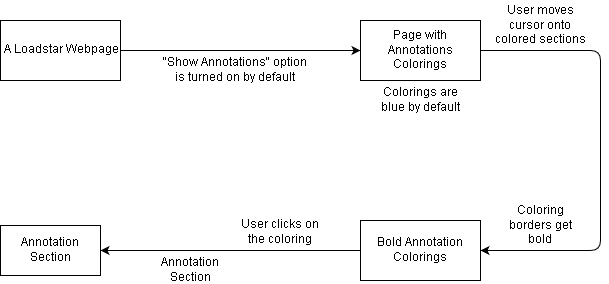
1. **The Method** 
   1. Annotation Types and Managing Social Dilemmas

In the platform, there are three types of annotations, public,private, and group. For every user, it is default to see public annotations. For group annotations to be seen, users should belong to a group somehow. Logic of creating and joining to groups is not designed yet. Therefore, we will only mention on public and private annotations on this section.

Showing of public annotations **may be a problem** if number of users in the platform exceed a certain limit. Popular tables could be filled with many annotations and become polluted with annotation colorings (text highlights). It may be benefitial to implement a filter to show a few popular annotations by default and if the user desires to see them all, ‘show all annotations’ option for that page could be selected.

This could be considered as an N-person dilemma in some cases such as trolls or people using public annotations for their personal notings. To solve N-person dilemmas, it is a common way to implement **costs on actors**.[[12]](#footnote-12)In our case, implemented cost could be a hybrid metric that measures the user’s experience and contribution in the platform,[[13]](#footnote-13) including such features as the time passed since the creation of the account, number of tables added that that are liked by users or mods, number of annotations added that that are liked by users or mods and daily usage streak.[[14]](#footnote-14) Users which have more experience and contribution could annotate more within a day.

A figure for the mechanism on showing of annotations has been drawn below. The process described is, user using the platform in a normal manner and reading a page. As the user sees an an annotation highlighting, he/she may move the cursor on it to see the annotation. When this happens, annotation typing gets bold and when the user clicks on the annotation, the annotation section appears. In annotation section, annotations for selected target and nearby target are shown, with specifying their targets.



Clicking on an Annotation by the User[[15]](#footnote-15)

* 1. Interface & End-User Perspective

The annotation Inteface should be designed as simple as possible in order to improve accesibility and urge people to annotate more. However, even in cases where interface is such that simple, logic may be a bit complicated. Below, there is a figure to show annotating logic on the platform.

In the platform, users can annotate simply by selecting text, images, audio with cursor highlighting[[16]](#footnote-16). However, the main issue here is if user wants to detailize his/her target selection on an item after selecting a number of items.

(It is possible for an annotation target to have more than one fragments in a page. Therfore, selecting more than one items in a page is an achievable function. Check the Chapter 2,2 for more information.)

This can be examplified as a user selecting two videos which are located in the same page, and then wanting to specify the minutes and seconds in videos in order to make his/her targeting more accurate for each video. In this type of cases, there would appear an annotation icon above each separate item user selected, according to the DOM[[17]](#footnote-17), and then user could click on those in order to detailize targeting. To finish targeting, user could click on the “Annotate” button, which appears everytime user selects a text.

In the below figure, it is implied that if the selected target is a sole text, detailed annotation is not avalible by default. Reason for this is that, users are already avalible to make detailed annotation targetings on sole texts with selecting the relvant parts with the cursor. Therefore there is no need for detailed targeting on sole texts. User could use the button Ctrl or Cmd in order to select seperated parts on the page.

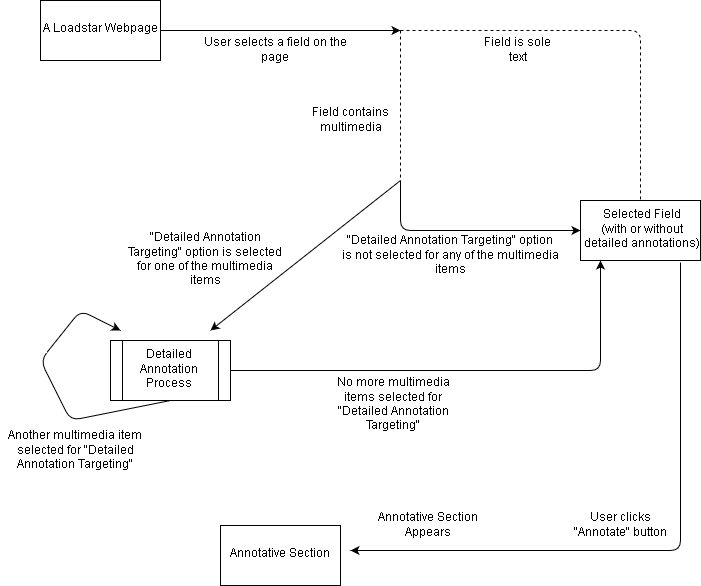


Figure describing the process of Annotation Targeting[[18]](#footnote-18)

After the targeting is done and “Annotate” button is clicked, Annotative Section appears. This section is to compose the the annotation body. Annotation body could include pictures in it (check Chapter 2.2), however videos are not to be stored inside annotations.

**4. Social Characteristics**

Loadstar is a table forming tool essentially, however, it needs to contain a certain amount of social activity in itself to fulfill its purpose. Annotation functionality is a great aid to the platform for boosting social activity. The benefits that annotations provide to the platform in a social sense can be listed as:

Accesibility, note taking, referencing, reviewing and detailization.

These are the benefits that provide individual users with a better experience. A broader perspective will be taken in Chapter 5, to see that all these benefits to individual users would provide the whole society with a better platform. However, in this section, perspective is narrowed down for the sake of simplicity.

* 1. **Accessibility**

As it is mentioned in section 1.2, accesibility is a key factor to reach all the people.

Annotations could provide:

* Visual content and audio content for people that have low-literacy.
* Audio content for blind people.
* Visual content for deaf people, when there is an audio content originally.
* Translations for people that are illiterate in a particular language.
* Alternative links for people that have internet limitations on their homelands.
  1. **Note Taking**

In Loadstar, annotations can be categorized as public, private, or group specific. While public annotations play a great role in accesibility, referencing, detailization, explanation etc, private and group specific annotations could provide a larger variety of benefits to specific user groups.

Note taking is one of these, mostly appealing to users on an individual level, as private annotations. Individuals could make their annotations private and take notes as they desire.

* 1. **Detailization**

Useful for both public, private and group specific annotations; providing and part of any page with further details.

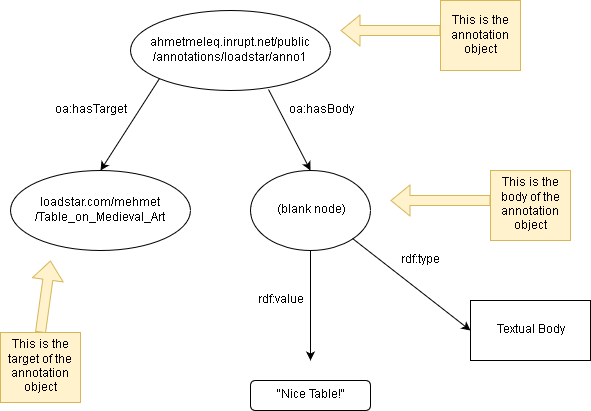
* 1. **Referencing & Reviewing**

Two important areas of usage espacially for academia. Mostly useful when annotation category is selected as group specific.

**5. Semantic Characteristics**

Annotations in Loadstar are fully semantic, due to Loadstar being based on Resource Description Framework itself, and Annotations being based on W3C’s standards. Semantic characteristics of Annotations in Loadstar could be listed as:

* Annotations are represented according to W3C’s Annotation Data Model. They are semantic items that connect targets and bodies. In a graph theory sense, they are acting as **bridges** to connect targets and bodies indirectly. Only function of an annotation object is that.
* Annotation relations are represented according to W3C’s Annotation Vocabulary. It will be easy for outsiders to benefit on the annotation data. Data will be easily included in **Linked Data.**
* Annotation Data will be stored in SOLID user pods. This will provide utility to traverse on Linked Data easier, using User Pods as bridge nodes.



It is possible to traverse further on the Linked Data in an automized way, provided that the root of the annotation object (User Pod) has connections that are irrelevant to the annotation.

**6. An Example Case**

In this section, an annotation example will be given such that:

* It covers a part of the Vocabulary.
* It covers some Important parts of data model and protocols.
* It will use user pods combined with the database of the platform for persistence.

Let’s say we have this table:

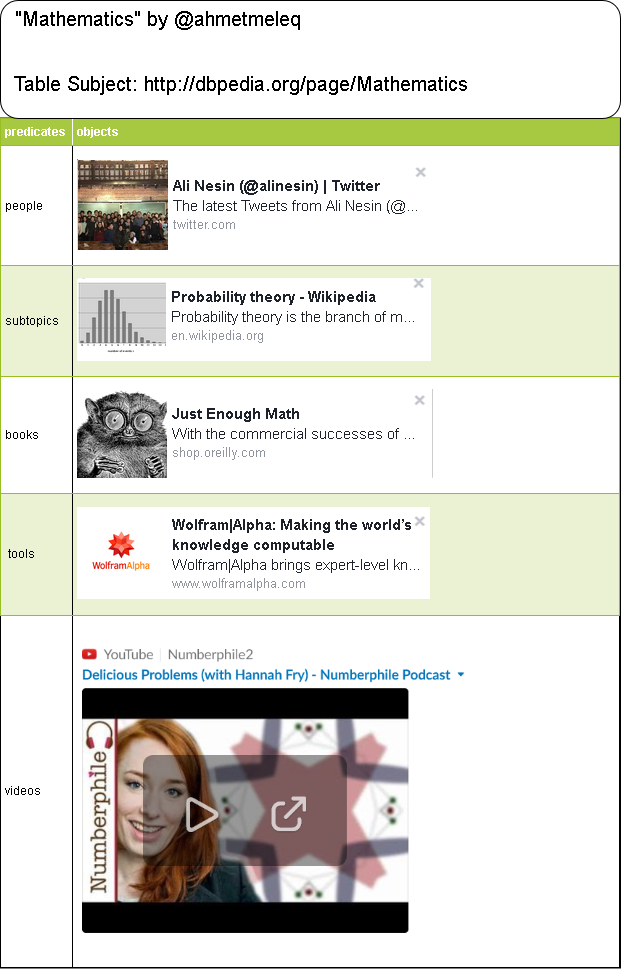
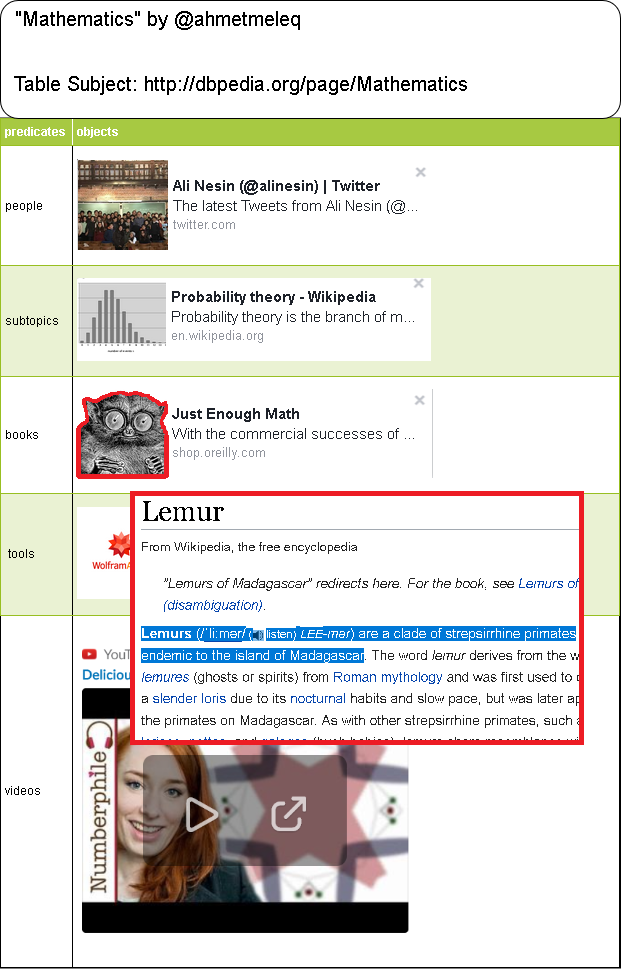


Table consists of 5 relations. In our case, the relevant one is the third row, Which has a book in it.

We want to annotate the picture inside the link box with a part of a Wikipedia page.

[[19]](#footnote-19)

Here are the steps for the user to annotate such content: (Our Method)

1. User selects the image (Lemur) with mouse cursor, holding the left mouse button and moving it across the image. This is similar to select text on a text editor.
2. User clicks on Detailed Annotation Targeting icon appearing on the picture selection.
3. Detailed Annotation Targeting Window Appears.
4. User selects the relevant part of the image with a SVG Selector tool.
5. User clicks on “Finish Detailed Targeting”
6. User clicks on “Annotate”
7. Annotative Window appears. (This is the window to compose the body of the annotation.)
8. User clicks on “Select body from a webpage” button.
9. A pop-up window appears.
10. User goes to the webpage (Wikipedia) which contains the annotation body.
11. User selects the relevant body inside the webpage.
12. “Add this to the Annotation Body” icon appears above the selection
13. User clicks on the icon.
14. User exits the pop-up window.
15. User now can see the relevant page with the selection info on the Annotative Screen. Selecting body from a webpage is succesful.
16. User clicks on “Finish Annotating “ button.
17. All the information is passed to backend and processed.
    1. Annotation is written on user pod if permission is granted beforehand. Else, annotation is stored inside one of the Loadstar Annotation Containers.
    2. If annotation is categorized as public, even if the user granted the Pod permission, a duplicate of the annotation object is stored inside Loadstar Annotation Containers.

Either in the Pod, or Platform’s Annotation Container; the Annotation would be stored as this :

(Vocabulary – Data Model)

<https://ahmetmelek.inrupt.net/Public/Annotations/Loadstar/anno1> a oa:Annotation ;

oa:hasBody [

oa:hasSource <https://en.wikipedia.org/wiki/Lemur> ;

oa:hasSelector [

a oa:TextQuoteSelector ;

oa:exact "Lemurs (/ˈliːmər/ (About this soundlisten) LEE-mər) are a clade of strepsirrhine primates endemic to the island of Madagascar." ;

oa:suffix " The word lemur derives from"

]

] ;

oa:hasTarget [

oa:hasSource <https://loadstar.com/ahmetmeleq/Mathematics> ;

oa:hasSelector [

a oa: XPathSelector;

rdf:value "/html/body/div[2]/div[2]/div[2]/table/tbody/tr[4]/td[2]/ul/li/div/div[1]/div"

] [[20]](#footnote-20)

oa:hasSelector [

a oa:SvgSelector ;

rdf:value "<svg:svg> ... </svg:svg>"

]

] . [[21]](#footnote-21)

Annotation could be serialized as this in JSON-LD format, which is a standard of Annotation Data Model by W3C. (Data Model)

{

"@context": "http://www.w3.org/ns/anno.jsonld",

"id": " https://ahmetmelek.inrupt.net/Public/Annotations/Loadstar/anno1 ",

"type": "Annotation",

"body": {

"source": " https://en.wikipedia.org/wiki/Lemur ",

"type": "TextualBody"

"selector": {

"type": "TextQuoteSelector",

"exact": “ Lemurs (/ˈliːmər/ (About this soundlisten) LEE-mər) are a clade of strepsirrhine primates endemic to the island of Madagascar."

“suffix”: “The word lemur derives from"

}

},

"target": {

"source": "https://loadstar.com/ahmetmeleq/Mathematics",

"selector": {

"type": "XPathSelector",

"value": “/html/body/div[2]/div[2]/div[2]/table/tbody/tr[4]/td[2]/ul/li/div/div[1]/div"

}

"selector": {

"type": "SVGSelector",

"value": "<svg:svg> ... </svg:svg>"

}

}

}

Annotation Container or the Pod would have to support these protocols as request to read the information:

GET, HEAD, OPTIONS

They would have to support these protocols as request to write information on them:

POST,PUT, GET, OPTIONS, HEAD, DELETE, PATCH

While creating the annotation, this request and response would take place: (Protocols)

Request:

POST /public/annotations/loadstar HTTP/1.1

Host: ahmetmeleq.inrupt.net

Accept: application/ld+json; profile="http://www.w3.org/ns/anno.jsonld"

Content-Type: application/ld+json; profile="http://www.w3.org/ns/anno.jsonld"

Content-Length: 874

Slug: “anno1”

{

-The JSON Serialization Here (it is shown previously)-

}

The slug is to suggest a URI to the Container.[[22]](#footnote-22) If it is accepted, Container returns a “Vary: Accept”

HTTP/1.1 201 CREATED

Allow: PUT,GET,OPTIONS,HEAD,DELETE,PATCH

Location: ahmetmeleq.inrupt.net/public/annotations/loadstar/anno1

Content-Type: application/ld+json; profile="http://www.w3.org/ns/anno.jsonld"

Content-Length: 874

ETag: "\_12a34bc567890"

Vary: Accept

{

-The JSON Serialization Here (it is shown previously)-

}

This POST method will probably be generated by the solid-auth-client library, since the library is able to login user accounts and track sessions. Here is a Node.js code with using solid-auth-client to generate the request:

solid.auth.fetch(

'https:// ahmetmeleq.inrupt.net/public/Annotations/load\_anno.ttl',

{

method: 'POST',

body: data

} )

.then(res => {return res;});

The annotation is posted, and able to get read. A similar request may be made to the Loadstar Annotation Container in order to store the annotation in platform’s database, if the annotation is public.

**7. Further Discussion**

To make queries related to the annotation and its creator, SPARQL query language may be used. The language is able to traverse Linked Data and bring results. This data could be visualized in order to view annotations links with bodies and targets. Other public data unrelated to the annotation could also be traversed, if the User has other public data in her pod.

Annotations could especially improve the expression of thought on the Web. Provided that there are mechanisms for people to make public annotations that everyone else can see on webpages, people will be able to express their thoughts in a much larger scale.

Annotations could also be used in a systematical way in order to help people on accesibility related issues.W3C’s recommendation provides relevant tools for this such as choice class which is benefitial to choose between different languaged versions of an annotation, or the accesibility relation which is to mention a accesibility feature.

Annotations also have a very interesting relation called motivation. This relation is to mention user’s intention while creating the annotation. This could be very beneficial for making sophisticated analyses with the Linked Data of Annotations, ie. runninga different script for every motivation, or filtering some annotations based on the motivation.

Annotations could be used to annotate random websites on the Web, given that those sites will leave a portion of their resources to strangers to make them leave their marks on the site. This would resemble the graffiti art on the urban areas. It could be a significant change in Web, if a number of popular websites would implement this.

**8. Conclusion**

Annotations are surely to change the Web, assumed that they will be included in attractive contexts. They form a powerful tool both for the user and applications. If users would have the strains on their hands, by the design of the data storage principles, their annotation data would be proven to be at safe hands, by design. This is why annotations are better to be stored in SOLID Pods.

Annotations could provide accesibility features, commenting tools, referencing tools for users and better analysing opportunities for organizations. Standards for annotations are provided by W3C. For the proliferation of annotation habit, tools that make annotating easy should be produced also. Both for developers and users, these tools are necessary for a web where annotations are often used to exist. These tools should not be profit-aimed nor commercialized. There should be libraries which are compliant with W3C’s recommendations, to make annotating easy on web servers, so developers could implement annotation tools on their sites and annotation habit could be spread among public. Until then, annotations will probably be used by academia by a majority.

In a future where every web site would provide every client with a limited amount of resource to annotate publicly, the web would surely be a much different place…

1. **References**

References are made on individual pages together with footnotes, here is a list of external sources that formed references:

* dbpedia.org (As an inspiration on basics of Loadstar.)
* https://www.w3.org/TR/2017/REC-annotation-model-20170223/#annotations
* https://solid.inrupt.com/
* https://en.wikipedia.org/wiki/Solid\_(web\_decentralization\_project)
* https://lod-cloud.net/
* https://www.w3.org/TR/2017/REC-annotation-protocol-20170223/
* https://github.com/solid/solid-auth-client/blob/master/README.md
* https://github.com/solid/solid-auth-client/issues/75
* https://www.w3.org/TR/2017/REC-annotation-vocab-20170223/
* https://www.w3.org/TR/2017/REC-annotation-model-20170223/
* https://www.w3.org/TR/2017/REC-annotation-protocol-20170223/#annotation-containers
* P. Kollock , Social Dilemmas: Anatomy of Cooperation, Annual Review of Sociology, 1998
* draw.io (To draw figures)
* facebook.com (To form link-boxes on figures, via Open Graph Protocol)
* slack.com (To form link-boxes on figures.)
* https://www.w3.org/TR/2017/REC-annotation-protocol-20170223/#suggesting-an-iri-for-an-annotation

1. This concept is inspired by Dbpedia. [↑](#footnote-ref-1)
2. https://www.w3.org/TR/2017/REC-annotation-model-20170223/#annotations [↑](#footnote-ref-2)
3. https://solid.inrupt.com/ [↑](#footnote-ref-3)
4. https://en.wikipedia.org/wiki/Solid\_(web\_decentralization\_project) [↑](#footnote-ref-4)
5. https://lod-cloud.net/ [↑](#footnote-ref-5)
6. <https://www.w3.org/TR/2017/REC-annotation-protocol-20170223/> [↑](#footnote-ref-6)
7. <https://github.com/solid/solid-auth-client/blob/master/README.md> [↑](#footnote-ref-7)
8. <https://github.com/solid/solid-auth-client/issues/75> [↑](#footnote-ref-8)
9. https://www.w3.org/TR/2017/REC-annotation-vocab-20170223/ [↑](#footnote-ref-9)
10. https://www.w3.org/TR/2017/REC-annotation-model-20170223/ [↑](#footnote-ref-10)
11. https://www.w3.org/TR/2017/REC-annotation-protocol-20170223/#annotation-containers [↑](#footnote-ref-11)
12. P. Kollock , Social Dilemmas: Anatomy of Cooperation, Annual Review of Sociology, 1998 [↑](#footnote-ref-12)
13. Such as in Wikipedia. [↑](#footnote-ref-13)
14. A scoring system to rank users on a basis of usage of the platform everyday without pauses. [↑](#footnote-ref-14)
15. Figure drawn via: draw.io [↑](#footnote-ref-15)
16. When you select with clicking and holding the mouse button, simply like selecting a text in a text editor. [↑](#footnote-ref-16)
17. Document Object Model [↑](#footnote-ref-17)
18. Figure drawn via: draw.io [↑](#footnote-ref-18)
19. Figure drawn via: draw.io [↑](#footnote-ref-19)
20. Inspired on DBpedia DOM and Facebook DOM with Open Graph Protocol Link Boxes. [↑](#footnote-ref-20)
21. Searched but could not find any guide on SVG selectors, could not examplify the values of SvgSelector. [↑](#footnote-ref-21)
22. https://www.w3.org/TR/2017/REC-annotation-protocol-20170223/#suggesting-an-iri-for-an-annotation [↑](#footnote-ref-22)