Fragalysis - Discourse Integration (WP2)

(For discussion)

Introduction

As part of the development of Fragalysis there is a requirement to link Fragalysis “ideas” with the external platforms Scarab and Discourse. This document covers the Discourse aspects.

The basic requirement is to implement the Discourse platform as part of the fragalysis stack and implement an endpoint or method to allow export of an idea to a discussion thread or topic.

This document estimates at a high level the work required to install the Discourse platform and add hooks to the fragalysis back end to post to Discourse when a user creates a *session project* or *snapshot* (representing an idea) from the Viewer.

There is a lot of functionality and plugins available in Discourse that can be configured directly. Further user actions could potentially be used to trigger posts in the future and other functionality such as upload/download links or events could be added. However to limit the complexity the scope will cover the session project and snapshot information only.

A similar set of basic functions will also be estimated for the Scarab integration for comparison.

Deployment

Discourse is already deployed to the STFC K8S cluster and is available at https://discourse.xchem-dev.diamond.ac.uk/

What is still needed is:
1. Configure authentication to use Keycloak for SSO. This looks to be straight forward as described here: https://meta.discourse.org/t/openid-connect-authentication-plugin/103632
2. Manage backups and restore
3. Configure email and other management functions
4. Deploy to production environment

Rough estimate of time involved: 2 days

Initial Use Cases:

Within Discourse the hierarchy of information is:

- Category
- *Subcategories* linked to a category
- *Topics* within a (sub)category
- and then *Posts* within a Topic.

We propose a simple relationship where a subcategory is created for each target and that this will be placed under a clear top level category called something like “Fragalysis targets”. Within each target subcategory, a Topic could logically relate to a session-project and posts could relate to snapshots (or ideas) within the Topic.

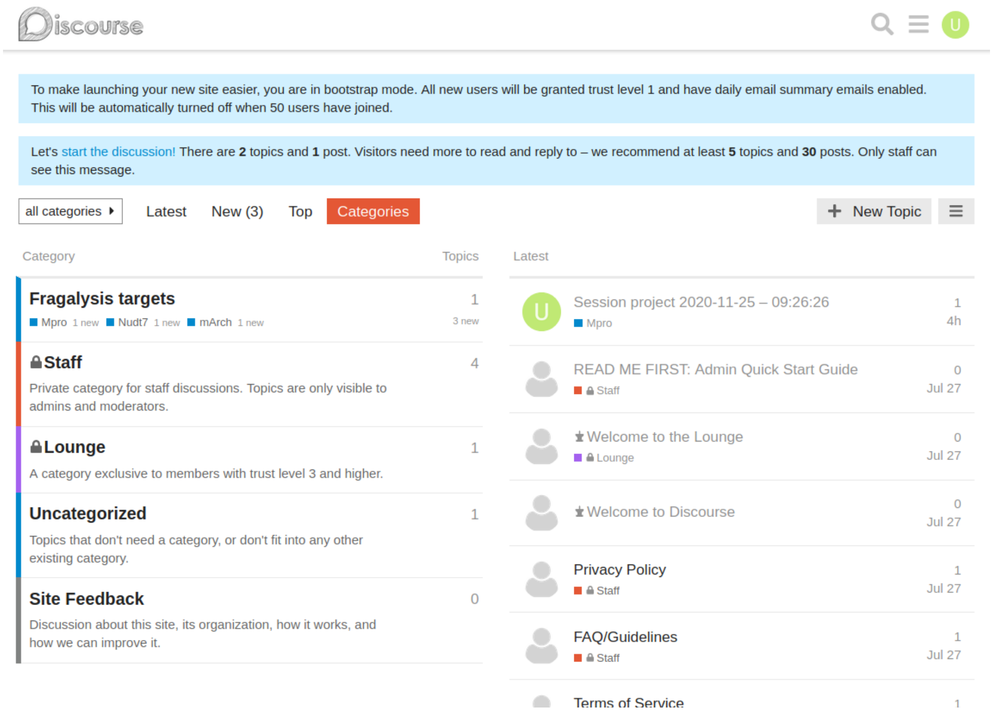
To demonstrate the processing, two initial use cases are imagined.

UC	Description
1	From the Fragalysis viewer, when saving a new session project, the user clicks on an option to also post to Discourse. A window opens and the user sees a window with contextual information (session project title, tags) and a box to add a comment (this box could be pre-filled with the description). When the user hits “post”, a call will be made using a new API (discourse_post) to the Fragalysis backend which will, in turn, call the Discourse API to: Set up a new subcategory for the target if not yet present Set up a new topic in the target subcategory with: - Title of the topic = session-project.title - Target recipients = {session-project.author} - Raw (content) = comment Invite recipients (based on Target Project)
2	From the Fragalysis viewer, when sharing a snapshot, the user clicks on an option to also post to Discourse. A window opens and the user sees a window with contextual information (session project title) and a box to add a comment (this box could be pre-filled with the description). When the user hits “post”, a call will be made using a new API to the Fragalysis backend which will, in turn, call the Discourse API to:

	Create a new post in the topic (idea) defined for the session project in the target subcategory with: - Target recipients = {snapshot.author} - Raw (content) = snapshot.description

This would produce an outline roughly like the following:

Top page: Categories selected



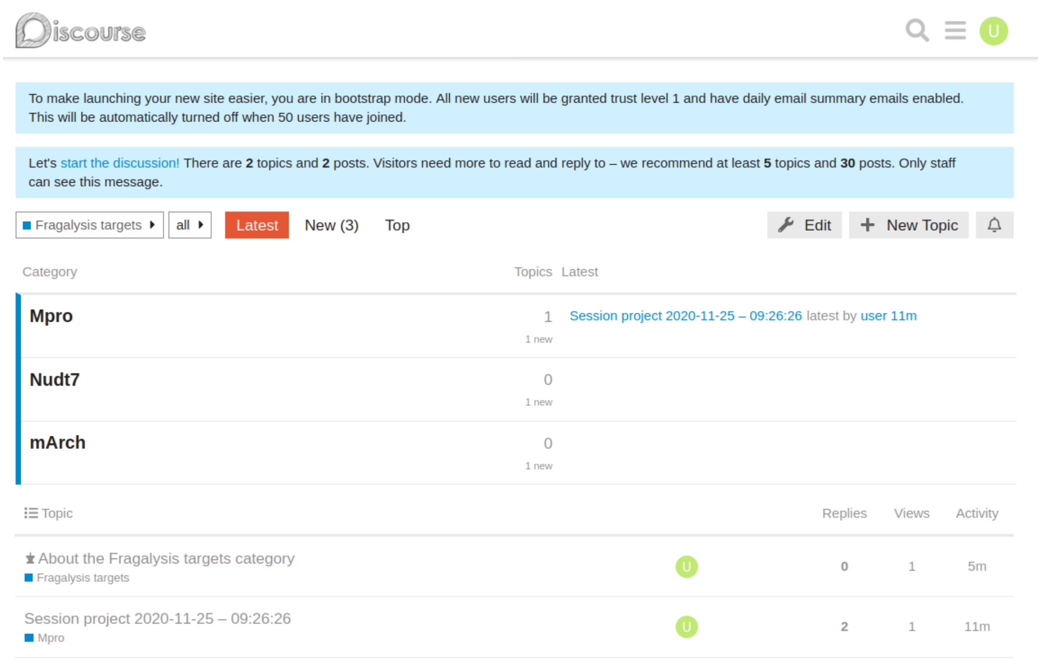
To make launching your new site easier, you are in bootstrap mode. All new users will be granted trust level 1 and have daily email summary emails enabled. This will be automatically turned off when 50 users have joined.

Let's start the discussion! There are **2** topics and **1** post. Visitors need more to read and reply to – we recommend at least **5** topics and **30** posts. Only staff can see this message.

Category	Topics
Fragalysis targets Mpro 1 new Nudt7 1 new mArch 1 new	1 3 new
Staff Private category for staff discussions. Topics are only visible to admins and moderators.	4
Lounge A category exclusive to members with trust level 3 and higher.	1
Uncategorized Topics that don't need a category, or don't fit into any other existing category.	1
Site Feedback Discussion about this site, its organization, how it works, and how we can improve it.	0

Latest		
Session project 2020-11-25 – 09:26:26 Mpro	1	4h
READ ME FIRST: Admin Quick Start Guide Staff	0	Jul 27
Welcome to the Lounge Lounge	0	Jul 27
Welcome to Discourse	0	Jul 27
Privacy Policy Staff	1	Jul 27
FAQ/Guidelines Staff	1	Jul 27
Terms of Service	1	

Within the Fragalysis targets

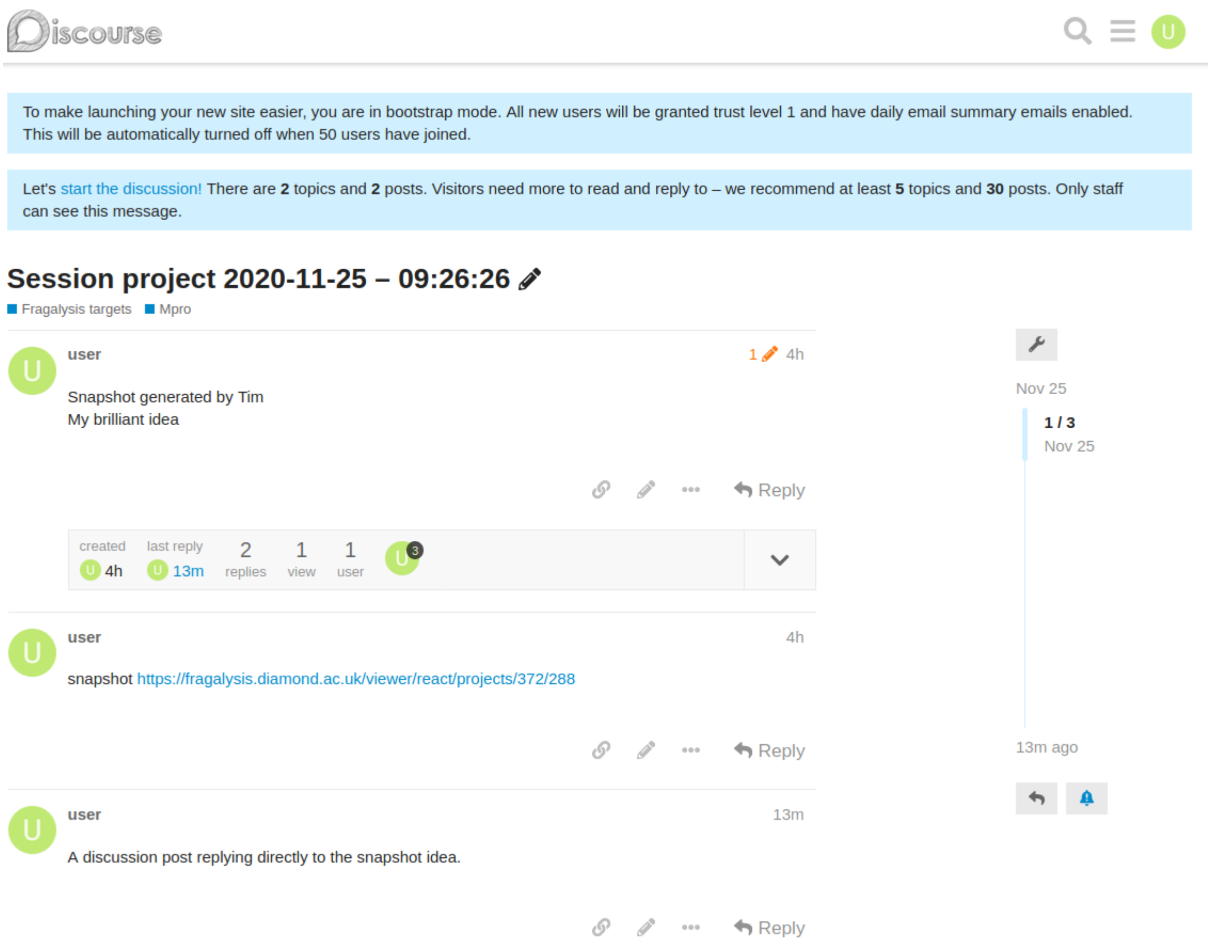

To make launching your new site easier, you are in bootstrap mode. All new users will be granted trust level 1 and have daily email summary emails enabled. This will be automatically turned off when 50 users have joined.

Let's start the discussion! There are **2** topics and **2** posts. Visitors need more to read and reply to – we recommend at least **5** topics and **30** posts. Only staff can see this message.

Category	Topics	Latest
Mpro	1 1 new	Session project 2020-11-25 – 09:26:26 latest by user 11m
Nudt7	0 1 new	
mArch	0 1 new	

Topic	Replies	Views	Activity
About the Fragalysis targets category Fragalysis targets	0	1	5m
Session project 2020-11-25 – 09:26:26 Mpro	2	1	11m

And within the project - note that this title is taken directly from the default on the “Save” popup - this can be customised.


To make launching your new site easier, you are in bootstrap mode. All new users will be granted trust level 1 and have daily email summary emails enabled. This will be automatically turned off when 50 users have joined.

Let's start the discussion! There are **2** topics and **2** posts. Visitors need more to read and reply to – we recommend at least **5** topics and **30** posts. Only staff can see this message.

Session project 2020-11-25 – 09:26:26

Fragalysis targets Mpro

user 1 4h

Snapshot generated by Tim
My brilliant idea

Reply

created 4h   last reply 13m   2 replies   1 view   1 user

user 4h

snapshot https://fragalysis.diamond.ac.uk/viewer/react/projects/372/288

Reply

user 13m

A discussion post replying directly to the snapshot idea.

Reply

Notes

1. Appearance/colours/icons are all very customisable - shown are just the defaults.
2. It seems that only **one** level of subcategory is available in Discourse - despite a lot of requests - so we can't go for a full tree structure. Maybe we can use tags for further links. See: https://meta.discourse.org/t/is-it-possible-to-create-sub-subcategories/13821

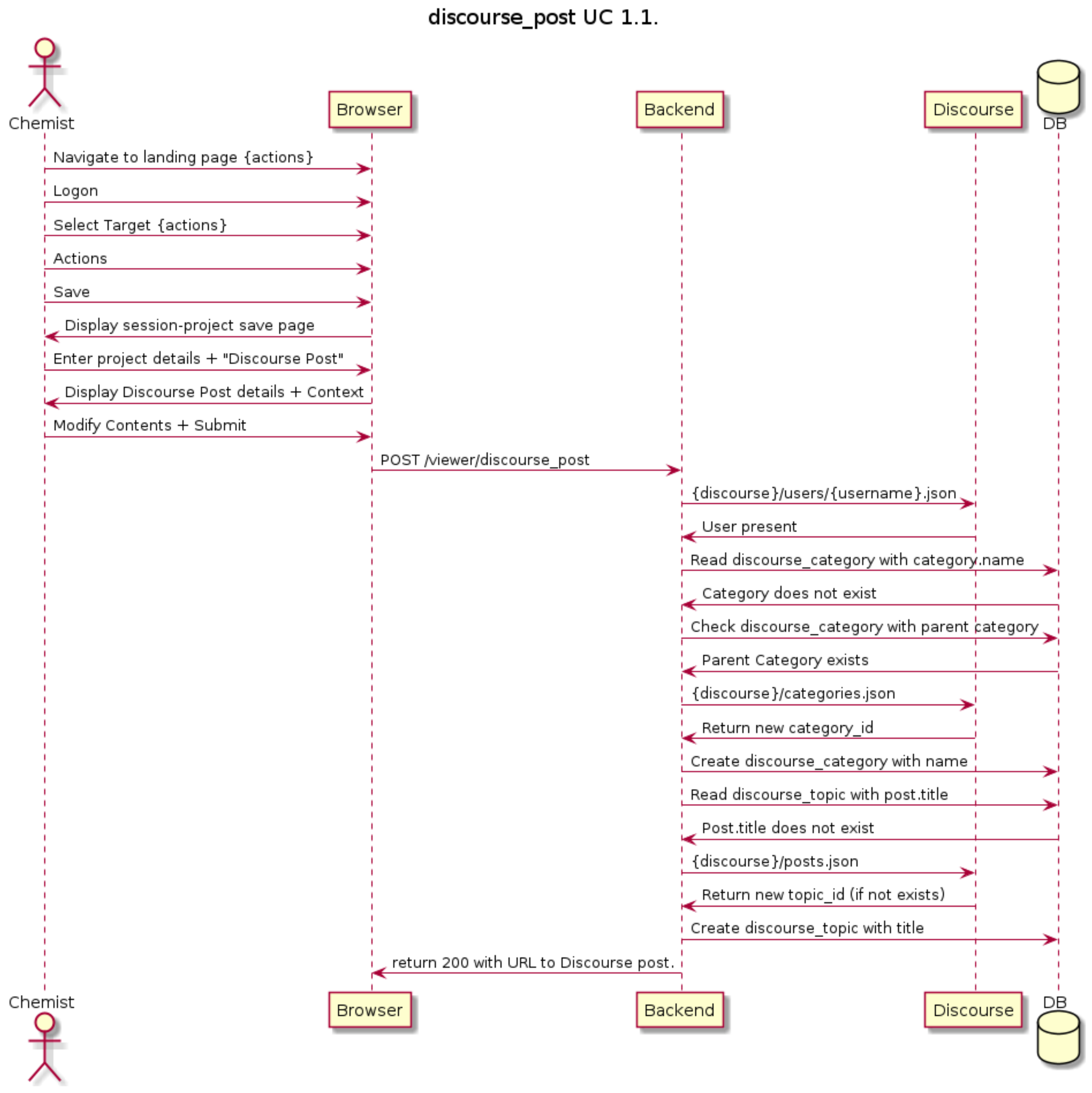
Solution Outline

Assumption: The post will be made in Discourse using the Fragalysis user name. Assuming that user creation in Discourse is handled externally, Fragalysis will assume that the user is created - so it will return an error if the user is not present in Discourse.

A hook will be created within the Fralgalysis Backend in form of a single that will call the relevant Discourse APIs. This hook will be called via a new Fragalysis API called *discourse_post* that will be made available to the frontend and will make the translation between fragalysis and Discourse - storing the reference information and making the multiple Discourse API calls necessary to allow:

1. Creation of a new (sub)category if not available
2. Creation of a Topic within a (sub)category.
3. Creation of a post within a topic.

The hook will be made as generic as possible so it can also be used for other purposes (as required by the front end). Hence the technical implementation below refers to categories, topics and posts rather than targets, session projects and snapshots.


discourse_post UC 1.1.
Chemist
Browser
Backend
Discourse
DB
Navigate to landing page {actions}
Logon
Select Target {actions}
Actions
Save
Display session-project save page
Enter project details + "Discourse Post"
Display Discourse Post details + Context
Modify Contents + Submit
POST /viewer/discourse_post
{discourse}/users/{username}.json
User present
Read discourse_category with category.name
Category does not exist
Check discourse_category with parent category
Parent Category exists
{discourse}/categories.json
Return new category_id
Create discourse_category with name
Read discourse_topic with post.title
Post.title does not exist
{discourse}/posts.json
Return new topic_id (if not exists)
Create discourse_topic with title
return 200 with URL to Discourse post.

Discourse_post

Create a post for an (optional)Topic for an (optional) (sub)category (which will typically correspond to a target name). This intended to be called from the React frontend when the user selects to make a post to discourse.

POST ../api/discourse_post/ - Create a post for an (optional)Topic for an (optional) (sub)category as JSON string.

Fields are as follows:

- category_name=<e.g. target name, *required if post_title not present*>,
- category_colour=<hex> (optional defaults),
- catetogy_text_colour=<hex> (optional defaults),

- parent_category_name=<parent category, *initially defaults to "Fragalysis targets"*>
- post_title=<e.g session-project.name, *required if category_name not present*>
- post_content=<e.g session-project.description + comment + URL, *required*>
- post_tags=<e.g. JSON list of tags, *optional*>

On a successful post, the URL to the discourse entry will be returned.

UC	Action	Parameters	Outcome (Discourse)
1.1	Upload new Target Save session project plus Initial snapshot	target.name Session-project.title	New subcategory New Topic
1.2	Existing Target Save session project plus Initial snapshot	target.name Session-project.title	New Topic attached to existing subcategory
2	Save/Share snapshot (idea) within session project	Session-project.title	New Post within existing Topic

Notes:

1. Numbers 1.1 and 1.2 are both covered in Use Case 1.
2. Subcategory names and Topic titles **have to be unique** within Discourse.
3. The user should be logged in to use this API (so that the correct user name can be sent to Discourse). An Authentication check will be made in the API.
4. For the front-end, to avoid errors, it would be good to add checks on the topic title to ensure that it's not less than 15 characters and to the content (raw) to ensure that it is not less than 20 characters. These are existing validations in Discourse.

Translation/Mapping to Discourse APIs.

Discourse translates the subcategory names/Topic titles into unique identifiers that are returned when the category or Topic are initially created and have to be supplied in future API calls. The fragalysis backend will store the identifiers and add them into the message for subsequent posts to existing categories/Topics.

The discourse_post API will initially make use of two Discourse API calls.

Categories.json - when it needs to create a (sub)category
(see https://docs.discourse.org/#tag/Categories/paths/~1categories.json/post)

Posts.json - when it needs to create a Topic or Post to a topic within a subcategory
(see https://docs.discourse.org/#tag/Posts/paths/~1posts.json/post)

Two new tables will be created:

discourse_categories

- category_name (unique)
- discourse_category_id

Note that this will store both the parent categories and subcategories.

discourse_topics

- topic_title (unique)
- discourse_topic_id

When the discourse_post api is called the fragalysis backend will:

1. If post details are present, check the category name is in the *discourse_categories* table. If the category name is absent in the table then retrieve the parent category id and call the categories API to create a new subcategory attached to the parent category id (see note 1 below). Store the category name and category id returned in *discourse_categories* for future posts.
2. If post details are present, check the topic name is in the *discourse_topics* table. If the topic_name is absent in the table - then call the posts API to create a Topic with the topic name (supplying also the subcategory id from *discourse_categories* if present - see note 2 below). Store the Topic name and Topic id returned in *discourse_topics* for future posts
3. If post details are present and the topic name is present in the *discourse_topics* table then call the posts API with the Topic id to attach the post to the topic.

Notes:

1. I'm assuming that the top level parent categories will be created in the *discourse_categories* table ahead of time. This can be worked out once the basic functionality works as desired.
2. I have deliberately not forced a category name to be present when creating a Topic. This is not strictly required in the Discourse posts API. If you don't supply a category then the Topic is created in the "uncategorized" category in Discourse.
3. The user-name attached to the post will be the logged in user (this is present in the Django database)

Summary:

1. New mapping tables (*discourse_categories* and *discourse_topics*).
2. Creation of discourse_post API
3. Testing of API
4. Documentation
5. Implementation notes (creation of parent categories) and any authentication/authorisation issues.

6. Integration with frontend (this will be a separate fragalysis frontend task).

Further Ideas/suggestions

Customization

1. Groups: Note that, in addition, within Discourse there is the option to restrict access to categories to certain groups of users
2. Notifications: Users can opt to be notified of various updates (new changes to Topics etc)
3. Icons/Fonts/Colours etc.

Add-ons that should be available (https://www.discourse.org/plugins) e.g:

1. Solved - stackoverflow-like functionality to choose an answer
2. Checklists
3. Calendar - create dynamic calendar from first post of topic
4. OAuth 2.0 & OpenID Connect Support.