Mnemosyne Flipboard-Like Application REST API

1. Introduction

1.1 Purpose

Client wants to develop a Hybrid mobile application built in HTML5/JS/CSS, which will be deployed to iOS and Android - Phone & Tablet sized.
This mobile application will aggregate RSS data from external sources, social media content, and internal content served via REST API, similar as Flipboard.
For internal content, we'd like to seek an existing content management sytem that fits our client's requirement to reduce the development effort.

This document lays out the contract for these RESTful services. Note that these services are actually RESTlike instead of RESTful.

2. Data Objects

2.1 FeedHeader

The FeedHeader object basically describes the feed. It is used to display the tile for a feed. JSON representation of the resource is as follows:

```
{
    "feed_id" : {long},
    "title" : {string},
    "link" : {url},
    "description" : {string},
    "image" : {url}
}
```

The data properties are summarized below:

Name	Type	Description
feed_id	Long	The id of the feed in our custom database.
title	String	The title of the feed. Required.
link	String	The URL corresponding to the feed. Required.
description	String	The description of the feed. Required.
image	String	The URL of the image for the feed. Optional.

2.2 ArticleHeader

The ArticleHeader object basically describes an article. It is used to display the tile for a article. JSON representation of the resource is as follows:

```
{
    "article_id" : {long},
    "title" : {string},
    "link" : {url},
    "description" : {string},
```

```
    "image" : {url},
    "author" : {string},
    "date" : {date},
    "category_id" : {long},
    "is_internal" : {boolean},
    "is_video" : {boolean},
    "total_dislikes" : {int},
    "total_likes" : {int},
    "total_comments" : {int}
}
```

The data properties are summarized below:

Name	Type	Description
article_id	Long	The id of the article in our database.
title	String	The title of the article. One of title or description is required.
link	String	The URL to the html website corresponding to the article. Required.
description	String	The description of the article. One of title or description is required.
image	String	The URL of the image for the article. Optional.
author	String	The Author of the article. Optional.
date	Date	The date on which the article was published
category	Long	The category id for the article
is_internal	boolean	whether this is an internal article
is_video	boolean	whether this is a video
total_likes	int	the total number of likes of the article. optional
total_dislikes	int	the total number of dislikes of the article. optional
total_comments	int	the total number of comments on the article. optional.

2.3 SearchResult

The SearchResult object is used in response of the search method. JSON representation of the resource is as follows:

```
{
    "articleLimit" : {integer},
    "articleOffset" : {integer},
    "feedLimit" : {integer},
    "feedOffset" : {integer},
    "articles" : [
```

```
        {ArticleHeader}, {ArticleHeader} …
    ],
    "feeds" : [
        {FeedHeader}, {FeedHeader} …
    ]
}
```

The data properties are summarized below:

Name	Type	Description
articleLimit	integer	The max number of articles to return. Default is 10
articleOffset	integer	The offset of articles from the start. Default is 0
feedLimit	integer	The max number of feeds to return. Default is 10
feedOffset	integer	The offset of feeds from the start. Default is 0
articles	Array of ArticleHeader objects	The articles that matched the search query
articles	Array of FeedHeader objects	The feeds that matched the search query

2.4 ArticlePagedResult

The ArticlePagedResult object is used in response of the search method. JSON representation of the resource is as follows:

```
{
    "pageNum" : {integer},
    "pageSize" : {integer},
    "totalItems" : {integer},
    "items" : [ {ArticleHeader}, {ArticleHeader}, ... ]
}
```

The data properties are summarized below:

Name	Type	Description
pageNum	integer	The page number It shall be positive integer, index is 1-based.
pageSize	integer	The number of items in a page.
totalItems	integer	Total number of result items if there were no paging.
items	Array of ArticleHeader objects.	List of ArticleHeader objects

2.5 Category

The Category object is describes a category in our DB. JSON representation of the resource is as follows:

```
{
    "category_id" : {long},
    "category_name" : {string},
    "category_image" : {url}
}
```

The data properties are summarized below:

Name	Type	Description
category_id	Long	The id of the category in our database. Required. PK
category_name	String	The name of the category. Required.
category_image	URL	The URL of the image for the category.

2.6 Article

The Article object is used to display the article details. It extends the ArticleHeader object and adds more fields to it. JSON representation of the resource is as follows:

```
{
    articleHeader: <articleHeader>,
    comments : [
        {
            id: {long},
            user_id: {string},
            user_name: {string},
            comment: {string},
            in_reply_to: {long},
            create_date: {date},
            likes_count: {integer},
            dislikes_count: {integer}
        },
        ...
    ],
    likes_count: {integer},
    dislikes_count: {integer},
    avg_rating: {float},
    related_articles: [
        <articleHeader>,
        <articleHeader>,
```

```
        ...
    ]
}
```

The data properties are summarized below:

Name	Type	Description
article_header	ArticleHeader	The basic article info
comments	Array	An array of all comments for the article, along with their likes and dislikes
likes_count	Integer	The number of likes for the article
dislikes_count	Integer	The number of dislikes for the article
avg_rating	Float	The average rating for the article
related_articles	Array	An array of ArticleHeader objects of related articles. The articles can simply be randomly selected list of articles from the same category and which are also somewhat recent. The maximum number of such articles to be returned must be configurable..

3. Services

All the date should follow the format “yyyy-MM-dd HH:mm:ss.SSS Z”.

The RESTful services will be hosted over a secure connection (https).

All methods will require user to be authenticated unless stated otherwise.

3.1 Login

This method is used to perform login into Mnemosyne.

Internally this method will call the Mnemosyne Login (which is out of scope). It is assumed however that the Mnemosyne login will return a userId, sessionId and sessionExpiresAt fields in JSON response. A custom wrapper will have to be likely written over the customer's current login process to return these fields.

The method will then save the login details to our mnemosyne_external DB.

- **URL**

  `https://<base-url>/login`

- **HTTP Method**

  POST

- **Request Description**

Name	Description	Required
`username`	The username	Yes.
`password`	The password	Yes.

- **Response Description**

Name	Description	Notes
`session_id`	A unique session id identifying the user.	The device must send this sessionId in all subsequent requests, to be considered logged in.

- **Notes:**
    1. Social Network Login will be implemented completely on the client side. The login will be performed at the client and it will store the access-token.
    2. The device must send this sessionId in all subsequent requests, to be considered logged in. The sessionId must be sent in the X-Auth-Session-Id custom HTTP header.

3.2 Logout

Logs out the user from the Mnemosyne network.

`https://<base-url>/logout`

- **HTTP Method**

  POST

- **Request Description**

Name	Description	Required
`sessionId`	Unique sessionId which was obtained during login.	yes

3.3 search (GET)

This method will be used to search articles by given parameters. There will be only one parameter called term. The search method will search for feeds and articles such that:

(a) The feed title starts with the given term

(b) The article title starts with the given term

(c) The article category starts with the given term.

(d) The article author starts with the given term

This method does not need authentication.

- **URL**

  `https://<base-url>/search?term=<term>&pageSize=<pageSize>&pageNum=<pageNum>`

- **HTTP Method**

  GET

- **Request Body**

  None

- **Response Returned**

  HTTP Status for successful response is 200.

  The response object will be a SearchResult object. See section 2 for all details on this object.

- **Request Description**

Name	Description	Required/Notes
`term`	The search term	yes
`pageSize`	The max number of items to retrieve in the search	no. defaults to 10
`pageNum`	The page number in the paged result to retrieve	no. defaults to 1

3.4 to read (GET)

This method will be used to get all articles that the currently logged-in user has marked as 'To Read'.

- **URL**

  `https://<base-url>/to_read?pageSize=<pageSize>&pageNum=<pageNum>`

- **HTTP Method**

  GET

- **Request Body**

  None

- **Response Returned**

  HTTP Status for successful response is 200.

  The response object will be a ArticlePagedResult object. See section 2 for all details on this object.

- **Request Description**

Name	Description	Required/Notes
`pageSize`	The max number of items to retrieve in the search	no. defaults to 10
`pageNum`	The page number in the paged result to retrieve	no. defaults to 1

3.5 to read (PUT)

This method will be used to mark an article as 'To Read' for the currently logged-in user. If the article is already marked as to-read then no change takes place.

- **URL**

`https://<base-url>/to_read/<article_id>`

- **HTTP Method**

  PUT

- **Request Body**

  None

- **Response Returned**

  HTTP Status for successful response is 201.

  The response will be empty.

- **Request Description**

  None

3.6 to read (DELETE)

This method will be used to unmark an article as 'To Read' for the currently logged-in user.

- **URL**

  `https://<base-url>/to_read/<article_id>`

- **HTTP Method**

  DELETE

- **Request Body**

  None

- **Response Returned**

  HTTP Status for successful response is 204.

  The response will be empty.

- **Request Description**

  None

3.7 categories (GET)

This method will be used to get all the categories in the system. Since the number of categories in the system are not assumed to be too large, the response will not be paged.

This method does not require user to be authenticated.

- **URL**

  `https://<base-url>/categories`

- **HTTP Method**

  GET

- **Request Body**

  None

- **Response Returned**

  HTTP Status for successful response is 200.

The response will be an array of Category objects. See section 2 for the definition of this object.

- **Request Description**

  None

- **Notes:**

  1. It is strongly recommended that the client app should cache the results of this method for a reasonably long time.

3.8 interests (GET)

This method will be used to get all the categories that the currently logged-in user has marked as Interest

- **URL**

  `https://<base-url>/interests`

- **HTTP Method**

  GET

- **Request Body**

  None

- **Response Returned**

  HTTP Status for successful response is 200.

  The response will be an array of Category objects. See section 2 for the definition of this object.

- **Request Description**

  None

3.9 save interest (PUT)

This method will be used to mark a category as Interest by the currently logged-in user.

- **URL**

  `https://<base-url>/interests/<category_id>`

- **HTTP Method**

  PUT

- **Request Body**

  None

- **Response Returned**

  HTTP Status for successful response is 201.

  The response will be empty.

- **Request Description**

  None

3.10 delete interest (DELETE)

This method will be used to unmark a category as Interest by the currently logged-in user.

- **URL**

  `https://<base-url>/interests/<category_id>`

- **HTTP Method**

  DELETE

- **Request Body**

  None

- **Response Returned**

  HTTP Status for successful response is 204.

  The response will be empty.

- **Request Description**

  None

3.11 favorites (GET)

This method will be used to get all the articles that the currently logged-in user has marked as favorite. The response will be paged.

- **URL**

  `https://<base-url>/favorites?pageSize=<pageSize>&pageNum=<pageNum>`

- **HTTP Method**

  GET

- **Request Body**

  None

- **Response Returned**

  HTTP Status for successful response is 200.

  The response will be an array of ArticlePagedResult objects. See section 2 for the definition of this object.

- **Request Description**

Name	Description	Required/Notes
`pageSize`	The max number of items to retrieve in the search	no. defaults to 10
`pageNum`	The page number in the paged result to retrieve	no. defaults to 1

3.12 favorites (PUT)

This method will be used to mark an article as favorite by the currently logged-in user.

- **URL**

  `https://<base-url>/favorites/<article_id>`

- **HTTP Method**

  PUT

- **Request Body**

  None

- **Response Returned**

  HTTP Status for successful response is 201.

  The response will be empty.

- **Request Description**

  None

3.13 favorites (DELETE)

This method will be used to unmark an article as favorite by the currently logged-in user.

- **URL**

  `https://<base-url>/favorites/<article_id>`

- **HTTP Method**

  DELETE

- **Request Body**

  None

- **Response Returned**

  HTTP Status for successful response is 204.

  The response will be empty.

- **Request Description**

  None

3.14 like article (PUT)

This method will be used to like an article for the currently logged-in user.

- **URL**

  `https://<base-url>/articles/<article_id>/likes`

- **HTTP Method**

  PUT

- **Request Body**

  None

- **Response Returned**

  HTTP Status for successful response is 201.

The response will be empty.

- **Request Description**

  None

3.15 dislike article (PUT)

This method will be used to dislike an article for the currently logged-in user.

- **URL**

  `https://<base-url>/articles/<article_id>/dislikes`

- **HTTP Method**

  PUT

- **Request Body**

  None

- **Response Returned**

  HTTP Status for successful response is 201.

  The response will be empty.

- **Request Description**

  None

3.16 rate article (PUT)

This method will be used to rate an article for the currently logged-in user.

- **URL**

  `https://<base-url>/articles/<article_id>/ratings`

- **HTTP Method**

  PUT

- **Request Body**

```
{
    rating: 3
}
```

- **Response Returned**

  HTTP Status for successful response is 201.

  The response will be empty.

- **Request Description**

Name	Description	Required/Notes
`rating`	The rating to give	yes. must be between 1 to 5.

3.17 comment on article (POST)

This method will be used to add comment to an article for the currently logged-in user.

- **URL**

  `https://<base-url>/articles/<article_id>/comments`

- **HTTP Method**

  POST

- **Request Body**

```
{
    comment: “Fully agree with you!”,
    in_reply_to: 12345876
}
```

- **Response Returned**

  HTTP Status for successful response is 201.

  The response will be empty. The response will contain a Location header with the generated comment_id in it.

  `Location: https://<base-url>/articles/<article_id>/comments/<comment_id>`

- **Request Description**

Name	Description	Required/Notes
`comment`	The comment to post	yes.
`in_reply_to`	The id of the comment to which this comment is a reply	no.

3.18 like comment (PUT)

This method will be used to like a comment for the currently logged-in user.

- **URL**

  `https://<base-url>/comments/<comment_id>/likes`

- **HTTP Method**

  PUT

- **Request Body**

  None

- **Response Returned**

  HTTP Status for successful response is 201.

The response will be empty.

- **Request Description**

  None

3.19 dislike comment (PUT)

This method will be used to dislike a comment for the currently logged-in user.

- **URL**

  `https://<base-url>/comments/<comment_id>/dislikes`

- **HTTP Method**

  PUT

- **Request Body**

  None

- **Response Returned**

  HTTP Status for successful response is 201.

  The response will be empty.

- **Request Description**

  None

3.20 view article (GET)

This method will be used to view an article

- **URL**

  `https://<base-url>/articles/<article_id>`

- **HTTP Method**

  GET

- **Request Body**

  None

- **Response Returned**

  HTTP Status for successful response is 200.

  The response will be an ArticleDetail object. See section 2 for the definition of this object.

- **Request Description**

  None

3.21 get articles for category (GET)

This method will be used get articles for category. This method does not need identification.

- **URL**

```
https://<base-url>/articles/<category_id>?
pageNum=<pageNum>&pageSize=<pageSize>
```

- **HTTP Method**

  GET

- **Request Body**

  None

- **Response Returned**

  HTTP Status for successful response is 200.

  The response will be an ArticlePagedResult object. See section 2 for the definition of this object.

  The total_likes, total_dislikes and total_comments fields for each ArticleHeader object must be set.

- **Request Description**

Name	Description	Required/Notes
`pageSize`	The max number of items to retrieve in the search	no. defaults to 10
`pageNum`	The page number in the paged result to retrieve	no. defaults to 1
`sortBy`	The field name by which to sort	no. defaults to pubDate
`sortDirection`	The direction of the sorting.	no. can be 'asc' and 'desc'. defaults to 'asc'

4. REST Error Codes

The error codes for the REST API are as follows:

Error Code	Description
401	If the SessionId is missing or not found in DB
400	If the request is invalid. This will be used where incoming arguments to REST method are invalid.
404	If the resource identified by the URL is not found.
419	If the SessionId is found in DB, but is expired
405	Method not allowed. If the REST URI is identified but the method requested is not supported for this URI.
500	An unexpected error on the server side happened.

The response will be a JSON object with a single field called “errorMessage”. This should be user friendly message as the mobile app will use it to display the message.