



# Journals Open Review Workflow and Integration

Google Summer of Code 2021

**NAME:** Apoorva Agarwal

**USERNAME:** [apoorva1509](#)

**EMAIL :** [apoorvaagarwal1509@gmail.com](mailto:apoorvaagarwal1509@gmail.com)

Hi I am Apoorva Agarwal, a second year student at Bits Pilani pursuing my degree in Mathematics.

My semester will finish in early May leaving me enough time to get ready for my GSoC project. If I am selected, I shall be able to work around 50-60 hours a week on the project, though I am open to putting in more effort if the work requires. I shall keep my status posted to my mentor on a weekly basis and maintain overall transparency in the project.

## **PROJECT PROPOSAL**

### **1. Overview:**

This project focuses on the CDLI journal workflow before we submit an article. Before the submission of journals it should be accessible for an open review process. To implement open journals in the system, an appropriate workflow needs to be developed which is intended to be carried out using [https://pkp.sfu.ca/ojs/ojs\\_download/](https://pkp.sfu.ca/ojs/ojs_download/) software. In addition to this, we intend to make further integration in the framework regarding endorsement of reviewers with published articles.

### **2. Milestone Details**

#### **2.1 Milestone 1 (Setting up ojs software in cdli framework with docker)**

To setup a new workflow for open journal system, we will use software [https://pkp.sfu.ca/ojs/ojs\\_download/](https://pkp.sfu.ca/ojs/ojs_download/).

OJS is an open source solution to managing and publishing scholarly journals online. OJS is a highly flexible editor-operated journal management and publishing system.

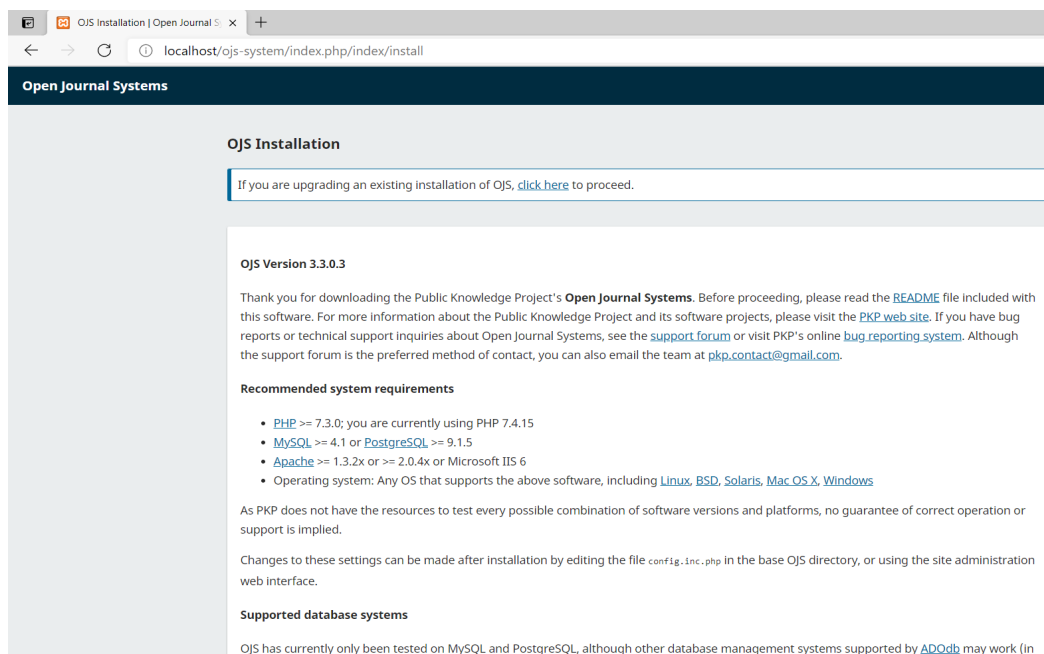
To download ojs,

#### System Requirements

To run the latest release of OJS 3.x, web server will need:

- [PHP](#) 7.3 or later with MySQL, MariaDB, or PostgreSQL support
- A database server: [MySQL/MariaDB](#) 4.1 or later OR [PostgreSQL](#) 9.1.5 or later
- UNIX-like OS recommended (such as [Linux](#), [FreeBSD](#), [Solaris](#), [Mac OS X](#), etc.)

- 1) Before installation, The [Docker Container for OJS](#) would be used and would be added to the Docker Container of the CDLI website.
- 2) The best solution to install ojs in our server is to install it using the [pkp package](#) which will be done inside the docker container made in the previous step. We will install this package in the app/ in the root directory.
  - For windows, we have to first install GNU patch(GNU Patch takes a patch file containing a difference listing produced by the diff program and applies those differences to one or more original files, producing patched versions.)
  - The published version of software will be in branches in gi3t repository which can be accessed using `git checkout stable-3_1_2`
- 3) Once ojs is downloaded, we will enable the system on a port and will integrate it in our framework wherever required.
- 4) After downloading ojs latest version,we will make a folder in **webroot/** dir `open_journals` to store all the articles and issues of cdli journals.
- 5) A new database will be created in sql server named `open_journals`.  
This is how installation page looks like on localhost:



- 6) After getting an idea on how to install ojs locally, I'll create a docker-compose file in the cdli framework (<https://docs.docker.com/compose/>) which would use alpine Linux, then I'll write the same commands which I used locally in the docker-compose.yml which will install ojs in the new container.

## 2.2 Challenges in Milestone 1

- **Docker Implementation:**

Creating a docker container for the ojs system needs to be done properly and we need to enable the ojs system on one port so that the ojs workflow runs parallel with our framework.

## 2.3 Milestone 2 (Building editorial workflow)

- 1) After installation, admin can login into the system and we will create a reader interface for our CDLI journal. The url of the open journals will lead readers and authors to a splash page like this. In the Dashboard of our OJS account, hover over the Navigation panel and choose Settings>Website .

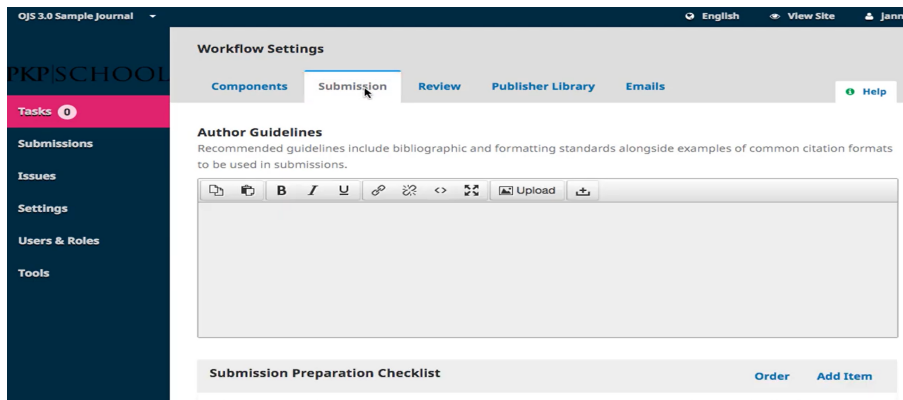
Refer [Fig 7b:Sample of Reader Interface in ojs](#)

- 2) We will create an editorial workflow which will be handle the following:
  - a. Submission of an Article
  - b. All the open review processes
  - c. Details of the article

### **Submission of Article**

In the Dashboard of our OJS account, hover over the Navigation panel and choose Settings>Workflow>Submission Tab. We can set various settings regarding submission of an article.

After that, the user will be able to login into his account and make a submission. This is how a submission screen appears to a user.



## Open review processes

Set up review in OJS 3

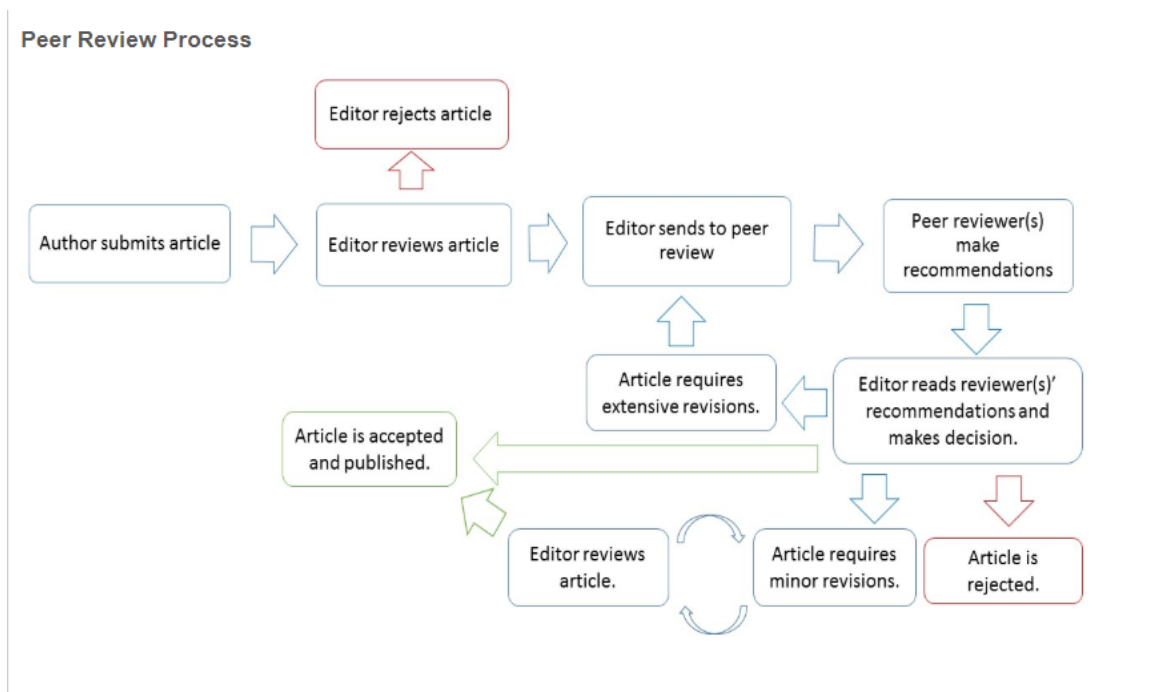
In the Dashboard of our OJS account, hover over the Navigation panel and choose Settings>Workflow>Review.

We can set up:

- reviewing deadlines
- automated email reminders for reviewers
- peer review type(Single-bind, Double-bind, Open)
- feedback form for reviewers

After we complete these settings, our open peer review workflow is set up and users can review our submissions.

The benefits of an open review process is that authors and reviewers are known to each other and encourage honest, accountable reviewing and persuade reviewers to do a thorough job.



All the Peer Review Process will be in our ojs dashboard. Once the review process is done we will publish it on our cdli dashboard.

## Details of the article

Our workflow will be able to list following details of the article:

- The document and its article history
- The original submission date
- The review date
- The major revisions
- The publication date
- Later revisions

Using the publications tabs after creating a workflow we can enable to list all the above details of the article.

The Publication tab allows us to edit or add information about the submission, including contributors, metadata, and identifiers. It is also where we can upload the final Galley files(like pdf , html) for publication in the journal.

**Workflow** **Publication**

Status: **Unscheduled** [Schedule For Publication](#)

**Title & Abstract** [Français \(Canada\)](#) [English](#)

**Contributors**

**Metadata**

**Identifiers**

**Galleys**

**Permissions & Disclosure**

**Issue**

**Prefix**  
Examples: A, The

**Title**  
Yam Diseases and its Management in Nigeria

**Subtitle**

**Abstract**  
B I x² x₂ [link](#)  
This article is about yam disease management in Nigeria

**Galley**

## **Some additional integration in the workflow:**

- **Citation Index(metrics):**

Our workflow will be able to show how to cite our article which can be done by adding the [Citation Style Language Plugin](#) and displays the number of downloads of an article on the article page which can be done by adding the [Usage statistics plugin](#).

- **DOI, ORCID, and PUBLONS integration:**

The DOI plugin under the Public Identifier Plugins is responsible for the assignment of the Digital Object Identifier to issues, articles, and galleys in OJS. It provides the user interface for DOI administration and assignment to publication objects. The plugin is part of the default OJS installation and does not have to be installed separately. To integrate it in our system we will follow [DOI Plugin Setup | Open Journal Systems](#) .

To integrate [ORCID \(Open Researcher and Contributor Identifier\)](#) in our system we will use the ORCID plugin. The ORCID Plugin Guide explains how to use and configure the ORCID Profile Plugin, how to obtain ORCID membership and API credentials, and how to connect ORCID iDs with works published with OJS. To integrate it in our system we will follow [Installation and Setup \(sfu.ca\)](#) .

A **Publons integration** lets our reviewers effortlessly track and verify every review they perform for our journal(s), without infringing on journal policies. The PUBLON plugin provides the ability to send and publish reviews to Publons (<https://publons.com>). To integrate it in our system follow [publons/ojs\\_3\\_plugin: OJS3 Plugin for the export of reviews to Publons \(github.com\)](#).

## **Publishing an article:**

Once the article is reviewed and accepted by the editor (ie the status of article is changed to be accepted), the particular article details will be exported from open\_journals database into cdli\_database and does publishing the article in [Publications - Journal \(ucla.edu\)](#) dashboard and removing it from ojs workflow.

## 2.4 Challenges in Milestone 2

- **Setting up Peer Review Process:**

Once the submission of an article is made, it should be accessed by all the users for open peer review. Before we start the open review process, we need to create a section on the website for review. A static page is needed to be designed for the peer review section which gives the user complete understanding of reviewing an article. In this section we will even try to add public comments. Besides reviewers can also suggest revisions.

- **Integrating details in the workflow:**

The additional features like citation, DOI or ORCID integration in the workflow requires installing plugins which need to be downloaded and integrated correctly in the OJS dashboard.

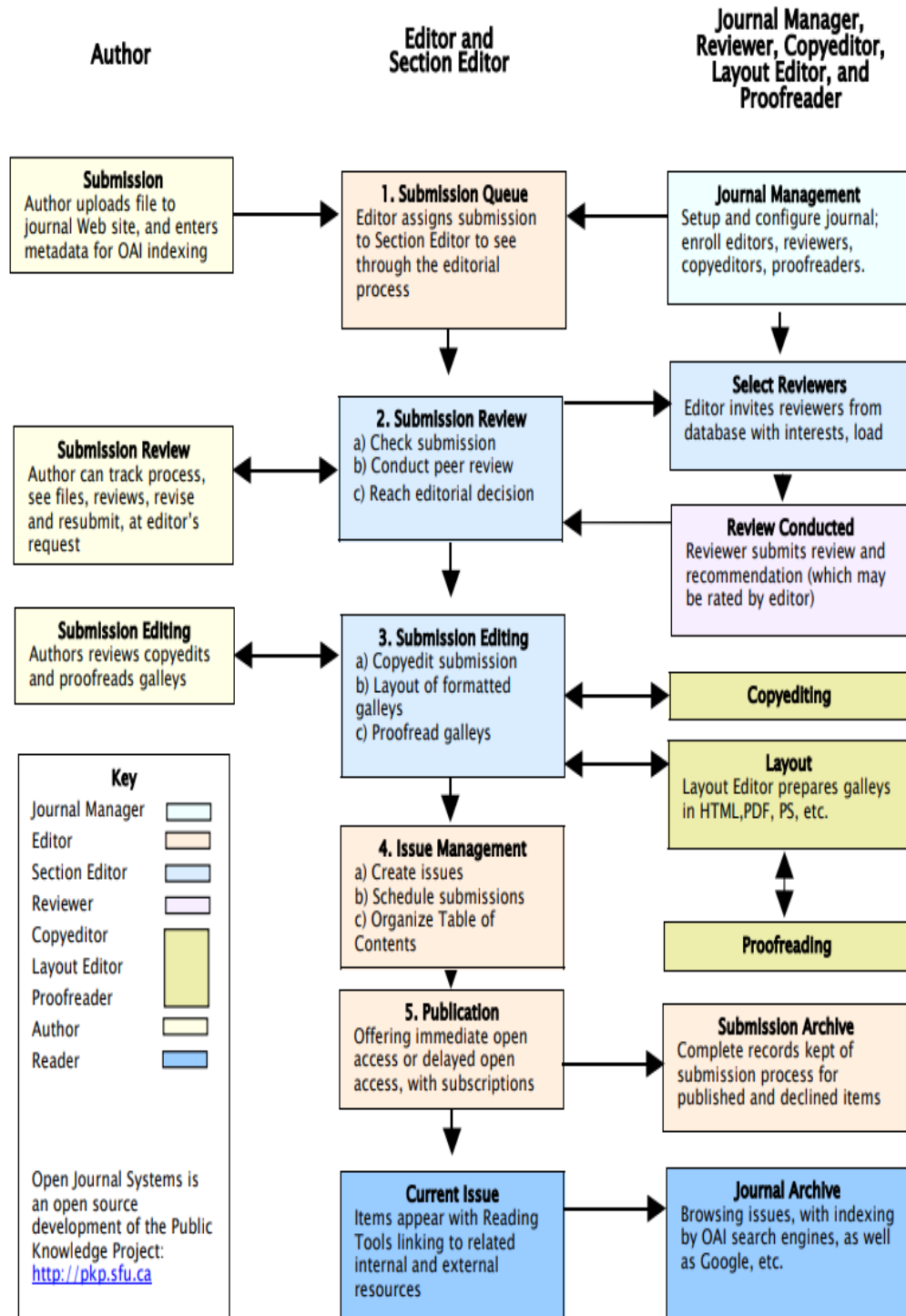
### **A Sample of OJS Workflow**

This is a sample showcasing some of the aspects of the ojs workflow.

1. **Submission** : When the author login to make a submission, he lands [here](#).
2. **Submission Queue**: After Submission, users can see all his submissions [here](#).
3. **Select Reviewers**: Add or select reviewer [here](#).
4. **Review Conducted**: Reviewer reviews the assigned submission [here](#).
5. **Submission Review**: Editors can conduct peer review [here](#).
6. **Author Submission Review**: Author reviews new revisions [here](#).
7. **Submission Editing**: Editor sends for copyediting [here](#).
8. **Copyediting**: After the editor submitted, copyediting can be done [here](#)
9. **Author Review Copyediting** [here](#).
10. **Issue Management**: After reviewing, create an issue and the issues can be created and managed [here](#).
11. **Publication**: Once the editorial workflow is completed, we will schedule the article for publishing [here](#).

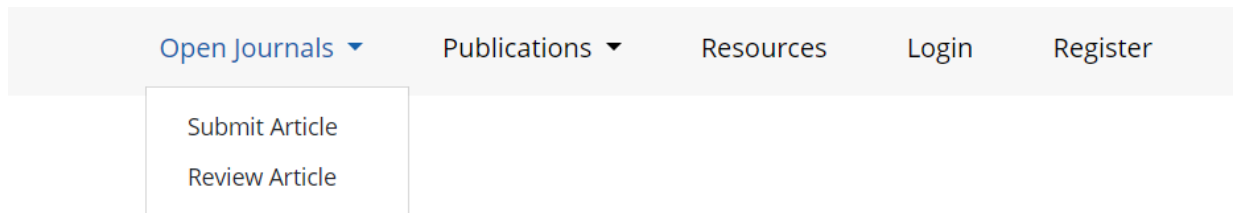


# OJS Workflow Chart



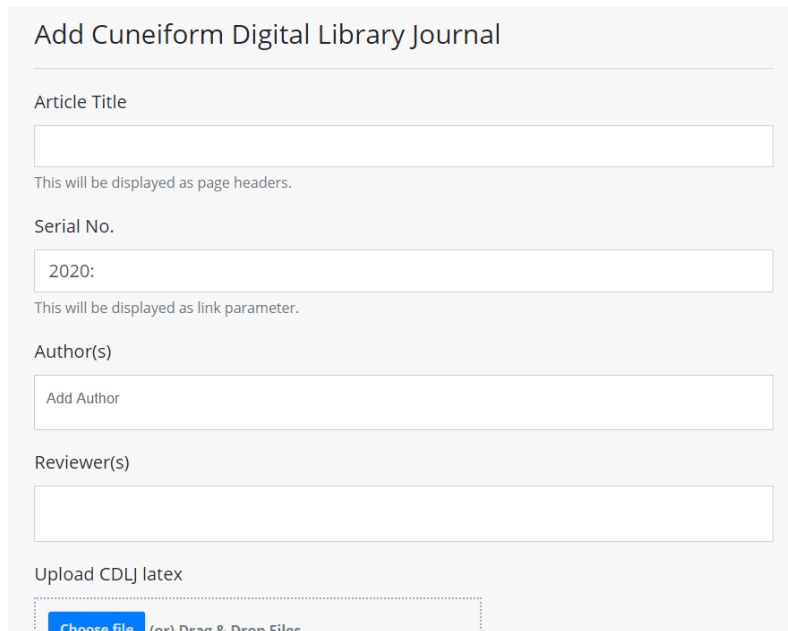
## 2.5 Milestone 3 (Integrating OJS Workflow in framework)

- 1) In Milestone 2 we have set up our open journal workflow. Now we need to integrate into our framework.



The Submit Article in Open Journals tab will redirect the page to submit articles in our new workflow to get peer reviews.

- 2) The reviewers will be linked to their authors profile. For that we will create a new entry in add/articles/cdlj. The reviewers will be added to the authors table in our cdli database so that we can link reviewers directly to their author profile.

A screenshot of a web form titled 'Add Cuneiform Digital Library Journal'. The form has several input fields: 'Article Title' with a text box and a note 'This will be displayed as page headers.'; 'Serial No.' with a text box containing '2020:' and a note 'This will be displayed as link parameter.'; 'Author(s)' with a text box containing 'Add Author'; 'Reviewer(s)' with a text box; and 'Upload CDLJ latex' with a file upload button labeled 'Choose file' and a note '(or) Drag & Drop Files'.

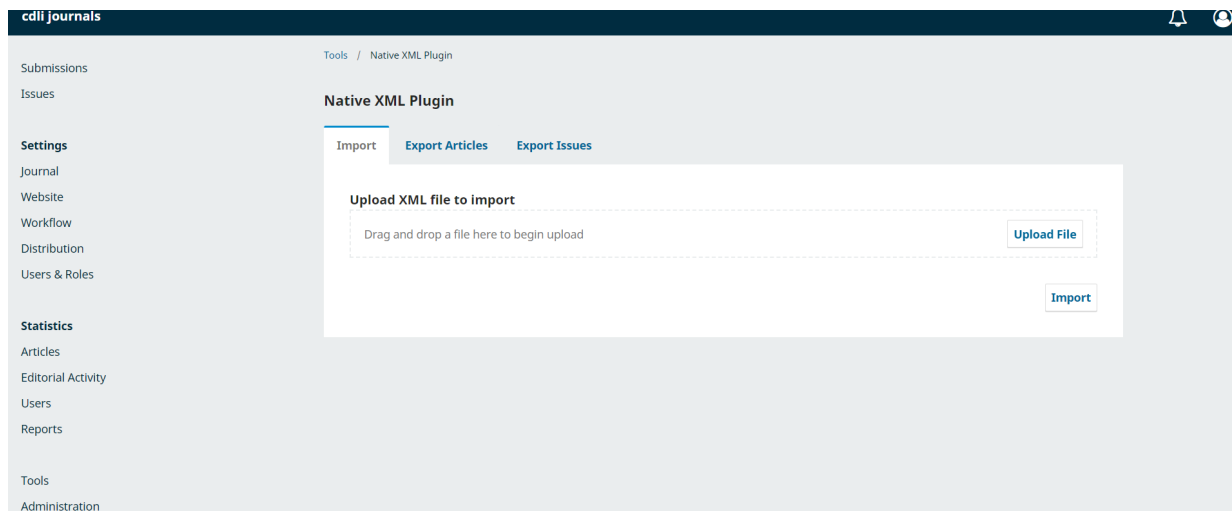
- 3) Once the ojs editorial workflow is ready, we need to set connection between two databases in our framework so that we can

- Submit an article from cdli journals dashboard for review
- Collect data from ojs workflow to publish final details of the articles with reviews on cdli journals dashboard.

## **Setting up connection between databases**

**Solution 1:** We will use the import/export plugin of ojs database. Import/Export allows you to easily get data out of your OJS journal and get data into it. Import/export functionality can be seen as a command-line tool [here](#).

We will use Native XML Plugin to export/import data between two dashboards. Hover over the Navigation panel and choose **Tools>Import/Export>Native XML Plugin**. It can be used to import and export single or multiple issues and/or articles, including comprehensive metadata. The details of this plugin and process to import and export can be found [here](#).



**Solution 2:**To connect both cdli\_db and open\_journals databases to our server we will add both databases like this in app/config/app.php.

Refer [Figure 7a: \(connect two databases dynamically \)](#).

After we have set up both the connections, we can use it for our ojs database in the framework in following ways,

- Schema: Within the Schema Builder, we can use the [Schema facade](#) with any connection. To specify which connection to use, simply run the connection() method:

```
Schema::connection('ojs')->create('some_table', function($table)
{
    $table->increments('id');
});
```

- Query: We can define a connection on a query builder

```
$users = DB::connection('ojs')->select(...);
```

- We can also define the connection at runtime via the [setConnection](#) method.

```
<?php

class ArticlesController extends ApplicationController {

    public function someMethod()
    {
        $someModel = new SomeModel;

        $someModel->setConnection('ojs');

        $something = $someModel->find(1);

        return $something;
    }
}
```

- 4) We will create a new static page in view page of published journal showing
  - a. The endorsement of reviewers with published articles.
  - b. (Some) review comments and final reviewer comments with published articles entries

A navigation bar(like shown below) will be added to the web template to switch between the article and review comments.



## **Database Schema:**

The database schema for the publications table has a few changes and thus after making these changes, data of the publication table from ojs database will be imported into the cdli database.

Appendices 1 publication database schema of ojs database

<u>publication_id</u>	<u>access_status</u>	<u>Date_published</u>	<u>Last_modified</u>	<u>locale</u>	<u>Primary_contact_id</u>	<u>section_id</u>	<u>seq</u>	<u>Submission_id</u>	<u>status</u>	<u>Url_path</u>	<u>version</u>
bigint	bigint	date	datetime	varchar	bigint	bigint	double	bigint	smallint	varchar	bigint

Some of the databases needed to be imported from ojs database into the cdli\_db are

Table name in ojs	Table Name in cdli(imported)	Primary Key	Description
Review_rounds	ojs_Review_rounds	article_id, round	Associates an article ID with a review file revision for each round of review
sections	ojs_sections	section_id	Defines sections within which journals can publish articles
published_articles	ojs_published_articles	pub_id	When an article is published, an entry in this table is created to augment information in the <b>articles</b> table
review_assignments	ojs_review_assignments	review_id	Stores information about reviewer assignments

A detailed idea of ojs database design can be found

<https://drive.google.com/file/d/15BK-KA8uzT0I9yrMEW2K9sISTXvX68Kc/view>

## **2.6 Challenges in Milestone 3**

- **Connection between two databases:**

We need to set up a connection between two databases, so that we can share data between them. Once the connection is set we need to update databases related to cdli journals so that we can import data from the OJS database and even import complete schemas from ojs database to cdli\_db. The connection should export articles from cdli journals for review to ojs workflow.

- **Publishing articles and details in cdli journals**

The various details of the article, html file and the endorsement of reviewers will be stored inside the cdli\_db which will be displayed directly as will be updated in the cdln\_web\_template.

## **2.7 Milestone 4**

### **Setting up testing pipelines and enhancements**

At the end of the first three milestones we will have a completely functional open review journal system, we will manually test the entire system in this milestone.

We will make a truncated copy of the current database and make a test-database configuration for the pipeline where we can start the php unit testing, for display of endorsement of reviewers.

We will also try to implement the lint tests when the desired project requirements are complete, we'll improve all the code styles to the latest coding trends and add the lint to the pipeline.

## **2.8 Challenges in Milestone 4**

### **Setting up the test database :**

The current CDLI database is heavy and has important details, hence we cannot use the same in the pipelines hence, the current database needs to be completely truncated and a testing database needs to be created. We will test import/export data first with the test database.

### **Add testing docs:**

When the necessary tests are created it is important to make a testing document so that other users can try running the code locally and make changes to the pipelines with confidence.

## **3. TIMELINE**

### **Phase 0: Pre-Community Bonding Period (Present - May 17th):**

- Fix the current issues present and make the setup process more smooth by creating a script for setup.
- Explore the framework and post new issues encountered.
- Explore more about OJS software and its implementation .

### **Phase 1: Community Bonding Period (May 17 - June 7):**

- Discuss with the mentor about the implementation in detail.
- Document the workflow for better understanding of the implementation approach.
- Determine all the dependent issues to be solved before getting started with the coding phase.
- Create a more detailed schedule with tasks and deliverable expected.
- Research more about open journal systems.
- Refactoring and cleaning up the code written by different applicants.

**Phase 2: Evaluating the milestone/Testing the flow (7th June - 30 th August):**

<b>Weeks</b>	<b>Tasks</b>	<b>Deliverable</b>
Week 1 (7th June-13th June)	<b><u>Milestone1</u></b> <ul style="list-style-type: none"> <li>● Inserting docker container of ojs</li> <li>● Download ojs software</li> </ul>	Setting up local environment to run ojs system
Week 2 - Week 4 (14th June- 4th July)	<b><u>Milestone2</u></b> <ul style="list-style-type: none"> <li>● Create an editorial workflow</li> <li>● Publishing article with review comments</li> <li>● Citation Index</li> <li>● DOI, ORCHID, and PUBLONS integration</li> </ul>	<ul style="list-style-type: none"> <li>● Submission of an article</li> <li>● Open Peer review process</li> <li>● Details and reviews of articles to be published</li> </ul>
Week 5 (5th July-11th July)	<ul style="list-style-type: none"> <li>● Test the new open journal workflow and complete the leftover</li> </ul>	<ul style="list-style-type: none"> <li>● Final testing of workflow</li> </ul>
Week 6 (12th July-17th July)	Manually test milestone 1 and milestone 2 and finish details for evaluation	
Week 7-Week 8 (19th July- 1st August)	<b><u>Milestone 3</u></b> <ul style="list-style-type: none"> <li>● Integrate ojs workflow in the framework</li> <li>● Setting up connection between databases</li> <li>● Import/Export Data</li> <li>● Make a section in the CDLI journals displaying the reviews</li> </ul>	<ul style="list-style-type: none"> <li>● Articles will be exported to ojs for review.</li> <li>● Articles will be published in cdli_journals</li> <li>● Reviewers should be linked with their author profile</li> <li>● Integrating endorsement of reviewers with published articles</li> <li>● Integrating (some) review comments and final reviewer comments with published articles entries</li> </ul>
Week 9-Week 10 (2nd August- 15th August)	<b><u>Milestone 4</u></b> <ul style="list-style-type: none"> <li>● Setting up the CI/CD with lint, a</li> </ul>	<ul style="list-style-type: none"> <li>● Make the first submission available</li> </ul>



	testing database will be created and a few unit testing as mentioned in milestone 4 for the journal dashboard. <ul style="list-style-type: none"> <li>Manually test the entire journal publication proposal, build few left out things and ready the project submission</li> </ul>	<ul style="list-style-type: none"> <li>Then either make anonymised reviews public but not names of reviewers, or make names of reviewers public but not reviews</li> <li>Make a full trail of peer review public</li> </ul>
Week 11(16th August-23rd August)	Manually test milestone 3 and milestone 4 and finish details for final submission.	

## **Contributions to cdli :**

I've contributed to cdli in the areas of journals dashboard, bugs , framework and design.

- changes made to (journal) articles indexes ([!269](#))
- Update button's CSS on (Admin Side) and added deactivate method for users ([#437](#))
- ck editor div has a scrollbar in add article ([#479](#))
- Changed day format of public journal indexes ( [!261](#))
- Remove static links from jquery.uploadfile.min.js ([#451](#))
- Change app debug to true in .env ([#448](#))

[Here](#) is a complete list of contributions made by me.

## **Future Plans (Post GSoC) :**

I have learned a lot of new things while contributing to the CDLI framework and even after Google summer of Code, I plan on making my contributions by working on open issues and enhancing dashboard features and open journal system workflow. Having picked up a good amount of developing skills, my major focus would be to develop good mentorship skills so that I can help other people navigate around and review their contributions.

## Why me?

Contributing to an open source organization seems very intriguing to me. I got to learn new skills, working with like-minded communities and challenging myself to implement design patterns and skills that couldn't be done before. I've been contributing to CDLI for a few weeks now and I'm overwhelmed by the way things are turning out. I've also made a significant amount of contributions, have been active on the discussions section on gitlab as well as slack.

## 4. DEV Details:

### **Timezone :**

I'll be in Indian Standard Time (IST) (UTC + 05:30) throughout the project timeline.

### **Contact Information :**

Apoorva Agarwal

Email : [apoorvaagarwal1509@gmail.com](mailto:apoorvaagarwal1509@gmail.com)

Phone : (+91) 7619844444

### **Profiles:**

Gitlab Username:- [apoorva1509](#)

Slack Username:- Apoorva Agarwal

Email Address:- [apoorvaagarwal1509@gmail.com](mailto:apoorvaagarwal1509@gmail.com)

Alternative Email Address:- [f20190597@goa.bits-pilani.ac.in](mailto:f20190597@goa.bits-pilani.ac.in)

Homepage:- [Apoorva Agarwal \(apoorva1509.github.io\)](#)

## 5. Appendices

### ● Appendices 1 publication database schema of ojs database

<u>publication_id</u>	<u>access_status</u>	<u>Date_published</u>	<u>Last_modified</u>	<u>locale</u>	<u>Primary_contact_id</u>	<u>section_id</u>	<u>seq</u>	<u>Submission_id</u>	<u>status</u>	<u>Url_path</u>	<u>version</u>
bigint	bigint	date	datetime	varchar	bigint	bigint	double	bigint	smallint	varchar	bigint

- **Appendices 2 Database Schemas needs to be imported into cdli\_db**

Table name in ojs	Table Name in cdli(imported)	Primary Key	Description
Review_rounds	ojs_Review_rounds	article_id, round	Associates an article ID with a review file revision for each round of review
sections	ojs_sections	section_id	Defines sections within which journals can publish articles
published_articles	ojs_published_articles	pub_id	When an article is published, an entry in this table is created to augment information in the <b>articles</b> table
review_assignment_s	ojs_review_assignments	review_id	Stores information about reviewer assignments

## **6. References**

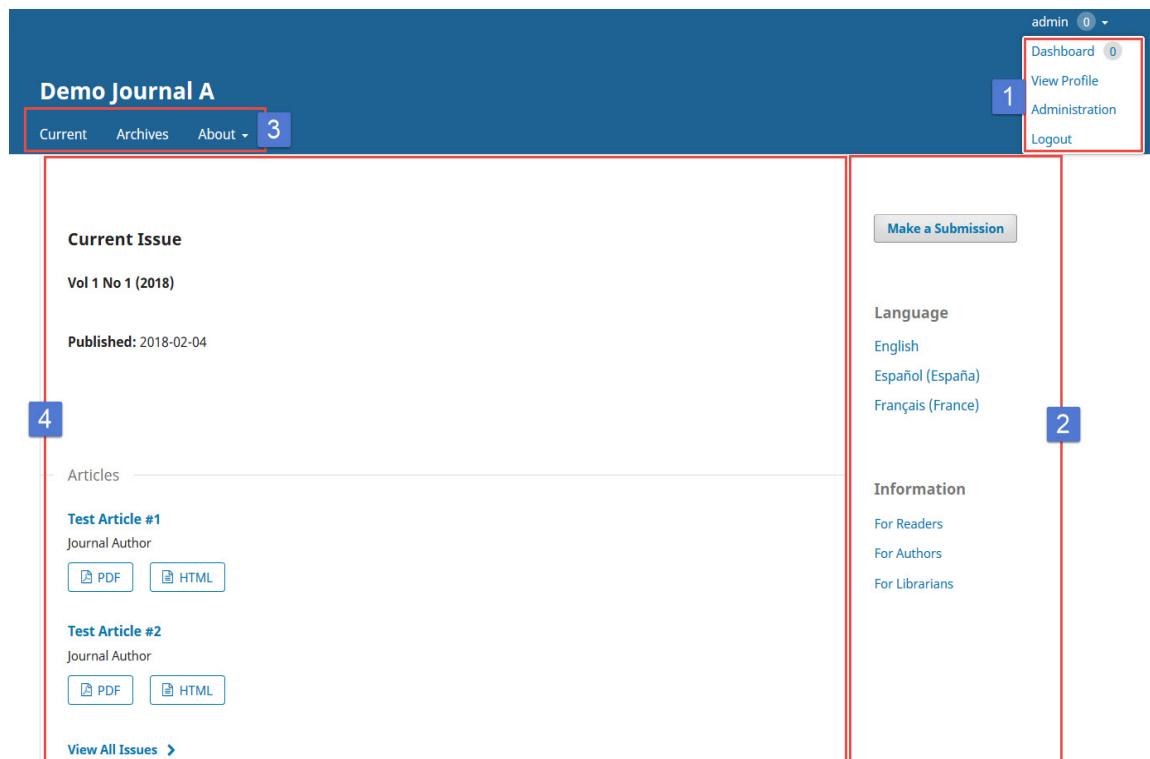
- <https://docs.google.com/document/d/1yuRujbVBZayKWm74Kvx7D-V2urWD7LT3jyqAah2YeGA/edit#heading=h.dq52tgvp2yjsx>
- [PKP School - Setting up a Journal in OJS 3 - YouTube](#)
- [OJS 3 User Guide | Open Journal Systems 3 User Guide](#)
- [Learning OJS 3.3: A Visual Guide to Open Journal Systems \(sfu.ca\)](#)
- [OJS Documentation \(sfu.ca\)](#)

## 7. Figures

*Fig 7a: Sample of connecting two databases dynamically*

```
'Datasources' => [  
  'default' => [  
    'className' => 'Cake\Database\Connection',  
    'driver' => 'Cake\Database\Driver\Mysql',  
    'persistent' => false,  
    'host' => 'mariadb',  
    'port' => '3306',  
    'username' => 'root',  
    'password' => '',  
    'database' => 'cdli_db',  
    'timezone' => 'UTC',  
    'flags' => [],  
    'cacheMetadata' => true,  
    'log' => false,  
    'quoteIdentifiers' => false,  
    'url' => env('DATABASE_URL', null),  
  ],  
  
  /**  
   * The ojs connection is used for ojs workflow.  
   */  
  'ojs' => [  
    'className' => 'Cake\Database\Connection',  
    'driver' => 'Cake\Database\Driver\Mysql',  
    'persistent' => false,  
    'host' => 'mariadb',  
    'port' => '3306',  
    'username' => 'root',  
    'password' => '',  
    'database' => 'ojs',  
    'timezone' => 'UTC',  
    'flags' => [],  
    'cacheMetadata' => true,  
    'log' => false,  
    'quoteIdentifiers' => false,  
    'url' => env('DATABASE_URL', null),  
  ],  
],
```

*Fig 7b: Sample of Reader Interface in ojs*



The screenshot shows the user functions from the profile menu at the top right of the screen [1]. Sidebar information is clearly broken out [2], as well as the top navigation bar with collapsible menus for the “About” functions [3]. Each article has a linked title for viewing object metadata and abstracts, and galleys are clearly labeled below the titles with clearer logos [4].