



Cuneiform Digital Library Initiative

Improving CDLI Framework

Proposal for Google Summer of Code 2020

Name : Vedant Wakalkar
Username : karna98
Email : vedantwakalkar@gmail.com
Location : Mumbai, India
Time Zone : UTC +5:30

1.Extend authorization/authentication system

The outcome of this challenge is essential to the core of the new CDLI Framework. It encompasses authentication and authorization, including access roles and granular access outside the generic permission roles.

The authentication with 2FA is already set up. But there are still bugs during registration like access needs to be properly configured for the roles.

Outcomes : When this challenge is complete, authentication will be set up in full. The access to pages which are already built will be set properly according to the appropriate level of access, and documentation will explain how to implement proper access to the new methods and views to be built in the future. The basis of an admin / editor dashboard should display links to restricted pages.

Issue 1: ByPassing 2FA (TwoFactorAuthentication).

Link : <https://gitlab.com/cdli/framework/-/issues/4>

The issue is related to 2FA (which has been fully set up) but still can be bypassed after registration. Whenever a user registers, they are redirected to page (URL : `'/twoFactor'`) to set up 2FA, but it can be skipped by navigating to different pages through the url.

Same in case of login, whenever the user has not enabled the 2FA, they should be redirected to the 2FA page and force them to set up.

For Login :

The problem can be solved by checking if the user has enabled 2FA or not.

```
/** In EveryController.php

// If login successful : set Auth["user"] = userDetails

// Checks if the user is logged in or not.
if ( authenticated ($User) ) {

    // Get the details of logged in users.
    $user = $this->Auth->identify();

    // Check if 2FA is enabled for user or not
    If ( $user["2fa_status"] == false ) {
        redirectTo('\twoFactor');
    } else {
        redirectTo('\home');
    }
} else {
    redirectTo('\login');
}
```

For navigating through URL :

Whenever a page is requested, the ***routes.php*** defines the routes to which controller should it invoke.

Controller is where all the **logic and data fetching and passing** it to the view happens.

There are two ways to solve the problem.

Approach 1:

Creating a **2FA check function** in every **controller**.

Whenever a **controller** is invoked, It will check if the user is authenticated or not.

As there are public pages and respective controllers, that means no authentication is required and hence no 2FA check.

But when a user is authenticated, it is mandatory for a user to have 2FA enabled hence we will check status of 2FA for authenticated users.

```
/** In EveryController.php

// Checks if the user is logged in or not.
// If authenticated the userDetails is stored in Session.
// Get the userDetails from Session
$user = $this->getRequest()->getSession()->read('User');

// Checks the 2FA status from $user["2fa_status"]
if (!$user['2fa_status']){

    // redirect to '/twoFactor'
    return $this->redirect([
        'controller' => 'Twofactor',    // TwoFactorController
        'action' => 'index'            // Index function of Controller
    ]);
}
```

Approach 2:

This approach is related to creating a middleware.

Adding a one more middleware layer to the default Routing.

Reference :

- 1) [CakeDC/auth/Authentication.md](#) (Recommended by CakePHP#Support)
- 2) [CakeDC/users/pull/439/files](#)
- 3) [How to add two factor authentication to your Laravel 5 application in 6 steps](#)

Approach 3:

Creating a check function as described in **Approach 1** in AppController.php.

AppController.php is used to defined application-wide methods in the class which can be inherited by all controllers.

Compared to all approaches,

Suitability for Implementation : Approach 2 > Approach 3 > Approach 1.

Approach 2 will be implemented for this issue taking security into consideration.

Issue 2: Setting up Role based Access System

Link : <https://gitlab.com/cdli/framework/-/issues/84>

To develop an authorization system on role based.

Different types of role present are :

1. Users
2. Editors
3. Granular not role related access
4. Admins

Granular access is miscellaneous access. Any user or Editor with an account can have all, none, or part of this access.

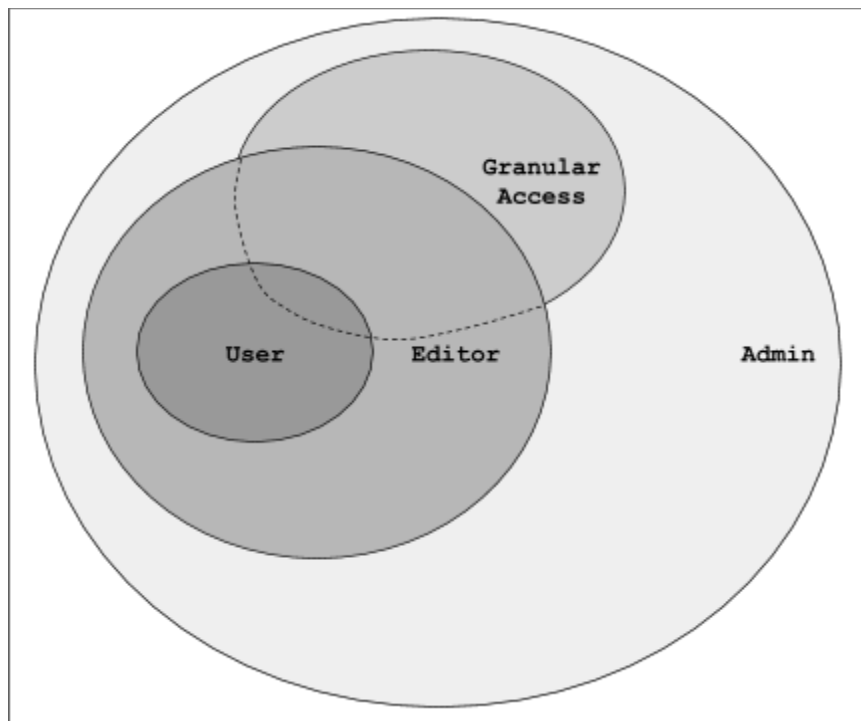


Fig. Roles Hierarchy

<u>Pages</u> Role	Admin Dashboard	Journal Artifacts	Artifacts & Catalogue	Users	Linguistic annotations & Transliterations	Images
User	No Access	View	View		View	View
Editor	Limited Access				Add, Edit	
Granular Access	No Access		Add, Edit, Delete, View (Private)		View (Private)	View (Private)
Admin	Full Access	Add, Edit, Delete	Add, Edit, Delete	Add, Edit, Delete	Add, Edit, Delete	Add, Edit, Delete

Fig. Overview of Permission for each Role

Approach 1:

In this approach, we can keep a checkRole function in Controller and display functionality according to the role of the Authenticated user.

Advantage:

1. Highly customizable
2. Single Controller for different roles.

Disadvantage: Manually have to write Condition for each and every controller.

Approach 2:

Creating an **Role Base Controller** (which is currently implemented).

The role has to be stored in a **Config file** with proper mapped access. **Policy(Rules)** need to be defined which will be initialized during the Authentication variable is created. Whenever a page is requested, in controller the user_role can be determined and policy will help to determine type of access for that role.

Advantage:

1. Policy defined makes it more secure and easy to use.

Reference :

- 1) [CakeDC/auth/Authorization.md](#)
- 2) [AlessandroMinoccheri/UserPermissions: a cakephp plugin](#)
- 3) [auth/Rbac.md at master · CakeDC/auth](#)

We will follow **Approach 2** for implementation of Role Based Access.

Additional Issues to be solved :

1. Strong Password checker

To implement a password checker is an important step to make password harder to crack and secure.

Rules for strong password :

- a. At least 8 characters—the more characters, the better
- b. At least one uppercase and lowercase letters
- c. At least one digit to be present.
- d. At least one special character, e.g., ! @ # ?]

Note: Avoid '<' & '>' in password, as both can cause problems in Web browsers

- e. Passwords should not contain username, first_name, last_name, etc..

```
bool validatePasswordStrength(String $password)    // to check if password is strong or not
{
    // Check length of password
    // Check for UpperCase Letter
    // Check for LowerCase Letter
    // Check for digit
    // Check for special Character
    // Check if contains username, name, etc.
}
```

Reference: [NIST's new password rules – what you need to know](#)

2. Implement Password Retrieval

Password Retrieval functionality has not been implemented yet. It will help users to retrieve their account who have forgotten their credentials.

Reference : [CakePHP-Auth-Forgot-Password/users_controller.php](#)

3. Fix '/logout'

This is a bug present in the current system. Users can add '/logout' to the base URL and the logged in user gets log out which is not expected behaviour i.e logout should be only done using POST request and not GET request.

If *logout* is invoked through the Logout button then the user gets logout. (POST)

If *logout* is invoked through url '/logout' (GET) then the user should be redirected to homepage or he should be displayed a warning box as shown below.



Reference : [Yet Other Examples of Abusing CSRF in Logout](#)

4. Edit Profile Page

Creation of Edit Profile page.

5. Proper Documentation of using 2FA with Google Authenticator

A proper guide to be documented to make users aware about the process of 2FA using Google Authenticator.

6. Making Account Inactive (After inactivity period more than 6 months)

Whenever a user tries to log in, his account details will be checked against the last activity for that account and status of that account (active or inactive). If the account has not been used for more than 6 months, the status of the account will be inactive and then the user will be sent a link to reactivate the account.

Periodically a function will be run to check if the accounts recorded last activity more than 6 months from present date, then their account status will be set to inactive.

```
bool updateAccountStatus()    // to check if password is strong or not
{
    $users = $this->getData('UserTables'); // Get users from UsersTable
    $currentDate = getNowTime();           // Get Current Timestamp
    for ($user as $users) {

        // Checking date difference greater than equal to 6 months
        if (dateDiffernece($user->lastActivity, $currentDate) >= 6) {
            $user->inactiveStatus = 1;    // Setting inactive status
            $user->save();                 // Saving Record
        }
    }
}
```

2.Finalizing main search and developing the advanced search

Outcomes: Arrays of results from simple and advanced search will give best results based on search query, custom rules and search features at a respectable short wait time.

Issue 1: Improving main search.

Link : <https://gitlab.com/cdli/framework/issues/37>

Main issue addressed here is making search queries efficient and displaying results on the page in pretty format.

Filtering can be done by client side processing using JavaScript (no query to backend), but there is possibility that JS code exposed to Client, can lead to security threat to the system. Hence we will keep the JS approach on hold and will follow the approach to query Controller for each search and filter.

1. ElasticSearch:

ElasticSearch is a distributed, open source search and analytics engine for all types of data, including textual, numerical, geospatial, structured, and unstructured. Adding ElasticSearch to our search module will make search more efficient and with less response time. The ElasticSearch will decrease the search time by 10 times.

- 1) Install ElasticSearch using composer.

```
composer install cakephp/elastic-search "^3.0"
```

- 2) Adding the plugin in **Application.php**

```
// Adding plugin ElasticSearch  
$this->addPlugin('Cake/ElasticSearch');
```

- 3) Then we have to configure the '**elastic**' datasource connection in **app.php**.
- 4) Creating an **Index** Object which can be used in SearchController .
- 5) Like ORM, ElasticSearch uses ODM which ORM like classes.
- 6) We then have to use ODM to define Document which will then index Documents.
- 7) Using ODM, documents get indexed and can be used to search using inbuilt queryBuilder of ElasticSearch.
- 8) Whenever new data is to be indexed, it has to be converted into **Document** then it is indexed.
- 9) Saving Document triggers :
 - a) Model.beforeSave
 - b) Model.buildRules
 - c) Model.afterSave
- 10) To get instance of index

```
// Getting index of Artifacts  
IndexRegistry::get('Artifacts');
```

Reference : <https://github.com/cakephp/elastic-search>

2. Search results options:

Whenever search results are displayed, users need to be provided with filter options to filter out results.

```
//View post data to Controller
{
  'filter' : $array(['filter_1']),
  'search_res' : $array([result])
}

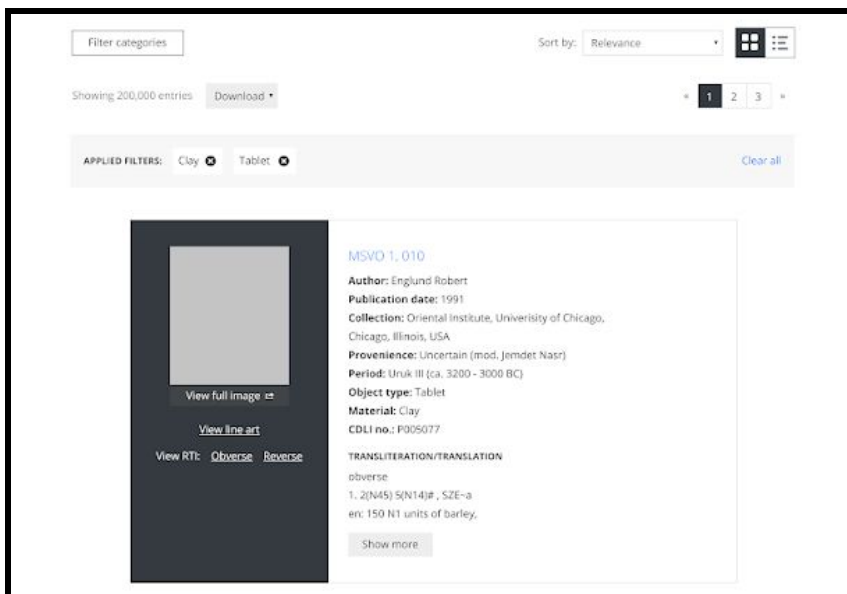
// In controller
$new_result = newSearchResultUsing(filter_1);

// Controller passes to View
{
  'filter' : $array(['filter_1']),
  'search_res' : $array([new_result])
}

// In View
Display->filter;
Display->search_res;
```

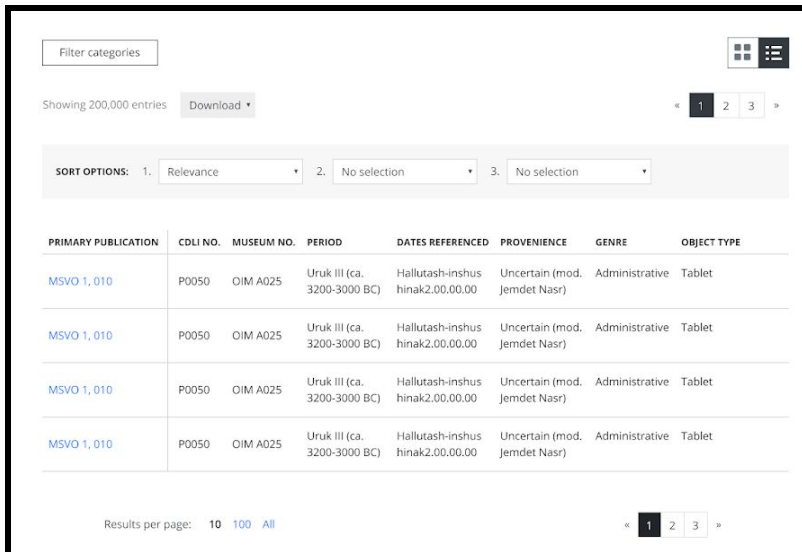
In view, when users apply a filter, a backend request is sent to the Controller. After the search query is filtered with the requested filter in Controller, the new search result is sent to View. An user can view the filtered result.

3. Full view options:



In Full View Option, the Search result is represented in more detail.

4. Compact view options:



PRIMARY PUBLICATION	CDLI NO.	MUSEUM NO.	PERIOD	DATES REFERENCED	PROVENIENCE	GENRE	OBJECT TYPE
MSVO 1,010	P0050	OIM A025	Uruk III (ca. 3200-3000 BC)	Hallutash-inshus hinak2.00.00.00	Uncertain (mod. Jemdet Nasr)	Administrative	Tablet
MSVO 1,010	P0050	OIM A025	Uruk III (ca. 3200-3000 BC)	Hallutash-inshus hinak2.00.00.00	Uncertain (mod. Jemdet Nasr)	Administrative	Tablet
MSVO 1,010	P0050	OIM A025	Uruk III (ca. 3200-3000 BC)	Hallutash-inshus hinak2.00.00.00	Uncertain (mod. Jemdet Nasr)	Administrative	Tablet
MSVO 1,010	P0050	OIM A025	Uruk III (ca. 3200-3000 BC)	Hallutash-inshus hinak2.00.00.00	Uncertain (mod. Jemdet Nasr)	Administrative	Tablet

In Compact View Option, the Search result is represented in less detail.

To display type of view (Full or Compact),

```
//In View.ctp
if (view_type == 'full') {
    Display (Image, Data1, Data2, Misc);
}
else { // view_type == 'compact'
    Display (Data1, Data2);
}
```

Issue 2: Advance Search Improvement.

Link : <https://gitlab.com/cdli/framework/issues/48>

In Advance Search, the previous work done is not working as expected.

In this issue we have to fix and improve AdvanceSearchController for making advance search work .

Since there are many filters, the complexity of query increases.

We have to rewrite the AdvanceSearchController with more efficient queries.

The main points to be considered :

- Searching much larger data.
- Custom Rules.
- Search Filter.

3.Prepare the search results display, expanded and compact

Outcomes : Expanded view of search results, compact search results for all 3 display formats, and downloads all set up properly and thoroughly tested.

Link : <https://gitlab.com/cdli/framework/issues/51>

Link : <https://gitlab.com/cdli/framework/issues/52>

The main issues are : providing an option to download the displayed data in various formats like

- a. Data preprocessing .
- b. Provide download options in various formats.
- c. Generate data in various formats like CSV, PDF, TTL, etc.

Data Preprocessing :

Whenever a user searches in Simple Search or Advanced Search, the result should be displayed with highlighted searched keywords. The processing has to be done in Controller and the processed result has to be sent to view.



Reference : <https://stackoverflow.com/a/10313526>

Generate data in various formats and providing download options for them:

After search results are displayed, each result consists of metadata, textual data and images. The user has to be provided with an option to download data as per requirement. The user can choose which type of data to be downloaded v.i.z. Metadata, textual data or images.

Metadata can be downloaded in

- a. Text (.TSV, .CSV) format
- b. PDF
- c. JSON-RDF
- d. XLSX
- e. TTL

Text format can be downloaded with user selected field delimiters. Field delimiters to be supported are tabs separated (.TSV) and comma separated(.CSV).

Textual data can be downloaded in

- a. ATF
- b. JSON-RDF
- c. CDLI-CONLL
- d. TTL
- e. PDF

Images will be of HD quality and will be presented in all web formats. They will be available for download in zip format.

The user can choose a combination among types of data to be downloaded and accordingly the download option will display the format available to be downloaded.

For ex. If a user wants to download textual data and image, he will be shown PDF to download.

Reference : [Understanding Linked Data Formats - Wallscope](#)

4.RocketChat Integration.

Link : <https://gitlab.com/cdli/framework/-/issues/95>

Using Slack has restricted the CDLI developers to most recent messages, as slack limits the users to 10,000 messages for community as it is a paid service. The alternative to Slack is RocketChat.

RocketChat is a free Open Source Solution for team communications.Setting up a developer community for CDLI will be of great help for future developers and will help to build a strong community. The RocketChat Integration can be done by building a container inside the docker and will initiate the RocketChat Server.

1) Configure the **docker-compose.yml**

```
version: '2'

services:
  rocketchat:
    image: rocketchat/rocket.chat:latest
    command: // Commands to start
    volumes:
      - ./uploads:/app/uploads
    environment: // Configure Environment Variables
    depends_on:
      - mongo
    ports:
      - 3000:3000

  mongo:
    image: mongo:4.0
    volumes:
      - ./data/db:/data/db
    command: // Initialize Mongo
```

2) Start the MongoDB

```
docker-compose up -d mongo
```

3) Starting Rocket.chat

```
docker-compose up -d rocketchat
```

4) Once the Rocket.Chat Container is running. Open Browser and set up the Admin Credentials.

5) In future whenever Rocket.Chat have to be updated to newer version, then follow

```
docker pull rocketchat/rocket.chat:develop
docker-compose stop rocketchat
docker-compose rm rocketchat
docker-compose up -d rocketchat
```

Reference :

1. <https://rocket.chat/docs/installation/docker-containers/>
2. <https://blog.ssdnodes.com/blog/tutorial-rocket-chat-docker/>

Timeline :

Phase 0: Pre-Community Bonding Period (Present - May 4th):

- Fix the current issues present (error generated during setup ex. py2-tools,Fuseki server) and making the setup process more smooth by creating a script for setup.
- Explore the framework and post new issues encountered.
- Explore more about Cakephp and its implementation .

Phase 1: Community Bonding Period (May 4 - June 1):

- 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 ElasticSearch.
- Refactoring and cleaning up the code written by different applicants.

Phase 2: Coding Phase 1 (1st June - 3rd July):

Week #	Task	Deliverable
Week 1 (1 st June - 7 th June)	<ul style="list-style-type: none">● Creating a middleware to redirect users not having 2FA enabled.● Testing Login and Registers with 2FA middleware.● Setup Password Strength Checker.● Develop Password Retrieval Module.● Documentation of Implementation.	<ul style="list-style-type: none">● Full functionality of login with password strength checker and password retrieval developed.● Developing 2FA middleware to force to enable 2FA.● Document 2FA middleware.
Week 2 (8 th June - 14 th June)	<ul style="list-style-type: none">● Creating a Configuration file with the roles mapped to access for each page.● Writing policy files with reference to Configuration.	<ul style="list-style-type: none">● Configuration file with user and editor role defined.● Modification of Auth.● Creation of Middleware.● Testing of Middleware.
Week 3 (15 th June - 21 st June)	<ul style="list-style-type: none">● Modify the \$this->Auth with adding user permissions, roles and respective access category.● Writing Middleware for Role based	<ul style="list-style-type: none">● Expanding Configuration file for Admin and other Granular access.● Testing Middleware for

	access. <ul style="list-style-type: none"> • Testing Module according to role hierarchy. • Testing Security check for unauthorized access. • Documentation of implementation of Access based control. 	Admin and Granular access. <ul style="list-style-type: none"> • Test files for all roles. • Final testing of Middleware for all roles. • Document Access based Control.
Week 4 (22 nd June - 3 rd July)	<ul style="list-style-type: none"> • Fix '/logout' functionality. • Create profile edit option • Create 2FA Google Authenticator Guide • Setting up triggers for inactive accounts. • Blog Post for Phase 1. 	<ul style="list-style-type: none"> • Fix all the additional issues as planned. • Report new issues generated. • Report for Phase 1.

Phase 3: Coding Phase 2 (3rd July - 31st July):

Week #	Task	Deliverable
Week 1 (3 rd July - 10 th July)	<ul style="list-style-type: none"> • Buffer week for remaining work in phase 1. • Fix additionally developed bugs. 	<ul style="list-style-type: none"> • Extend Authentication / Authorization System without any issues. • Document the remaining feature.
Week 2 (11 th July - 17 th July)	<ul style="list-style-type: none"> • Setting up ElasticSearch for Simple Search. • Modifying Search Result options. • Filter based searching • Implementing Full and Compact View Options. • Testing Full and Compact View for Search. • Creating Test for Simple Search. • Optimizing Search query. • Benchmark response time. 	<ul style="list-style-type: none"> • Elastic search setup for Simple Search. • Full setup of Simple Search. • Test files for Simple Search. • Documentation of implemented Simple search with ElasticSearch.
Week 3 (18 th July - 24 th July)	<ul style="list-style-type: none"> • Fixing UI of Advance Search page. • Rewriting AdvanceSearch Controller. • Optimizing Query. • Creating Test Files. • Benchmark response time. 	<ul style="list-style-type: none"> • Fully Functional Advanced Search. • Test File for Advance Search. • Documentation of implemented Advanced Search.

Week 4 (25 th July - 31 st July)	<ul style="list-style-type: none"> ● Buffer for any remaining work. ● Blog Post For Phase 2 	<ul style="list-style-type: none"> ● Analysis and Report new issues generated. ● Remaining Documentation (if Any) ● Report for Phase 2.
-----------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Phase 4: Coding Phase 3 (31st July - 31st August):

Week #	Task	Deliverable
Week 1 (31 st July - 7 th August)	<ul style="list-style-type: none"> ● Data processing for Search views. ● Setting up a download option for results. ● Generating different types of data formats. 	<ul style="list-style-type: none"> ● Highlight the searched strings as a result. ● Download option ready with all types of data formats and customization. ● Generating different types of Data. ● Documenting the different types of download option available and data formats requirements.
Week 2 (8 th August - 14 th August)	<ul style="list-style-type: none"> ● RocketChat Integration. ● Testing RocketChat ● Documentation ● Refactoring and cleaning up code ● Fixing bugs ● Update Documentation 	<ul style="list-style-type: none"> ● RocketChat ready to be deployed with a guide to use.
Week 3 (15 th August - 21 st August)	<ul style="list-style-type: none"> ● Buffer week for pending deliverables. 	<ul style="list-style-type: none"> ● TBD after discussion with my mentor.
Week 4 (22 nd August - 31 st August)	<ul style="list-style-type: none"> ● Prepare final Project Report. ● Submit the Code. 	<ul style="list-style-type: none"> ● Final submission of Report, Code and blog posts.

Future Plans (Post GSoC) :

Post GSoC, I will be active in the community and will try to contribute new features for CDLI Framework.

Contributions to CDLI Framework :

Type	Link	Description
Enhancement PR	PR 75	Fixed Register form validation message
Enhancement PR	PR 78	Fixed Empty fields Registration (Major Security Issue)
Enhancement PR	PR 81	Fixed Deprecated Error

Why are you the right person to work on this project?

- I have been working on Web Development for the past 2 years with Laravel (PHP based framework) and Spring Boot (Java based framework).
Since both follow MVC Architecture, hence I've a good understanding of architecture and its implementation.
- I started competitive programming during my freshman year and still enjoys it. During sophomore year, our team were top 10 teams in sPrdh (Departmental Project competition) which was 3 months long. Our project was to build an eCommerce website to bridge the gap between farmers and Government.
- During the summer of my sophomore year, I did my internship within my institute to build a Content Management System and Attendance System.
- My team were semi-finalists in the Deep Blue Project by Mastek and Majesco Pvt. Ltd.
- We won the National Level Hackathon Smart India Hackathon 2019 conducted by MHRD, Govt. of India for Goonj Organization.
- My teammates and I were in the top 10 teams in Mumbai Hackathon, which was an Open Source Project Hackathon. We built an [Offline Collaborative Editor](#) with basic functionality.
- I have prior experience of working on Open Source Project. I was one of the top 50 contributors during GirlScript's Summer of Code 2019.
- Recently I have completed [30 Days of Javascript](#) where I learnt new concepts of JavaScript every day for a month.
- I have tried contributing to Open Source Projects like Coala, [Algo_Ds_Notes](#). I was part of GirlScript's Summer of Code 2019, that's when I contributed to Algo_Ds_Notes by coding different algorithms in various programming languages so that it is easy to access for others to get the Algorithm in their preferred language.

Other commitments :

No prior commitments as of now. If I'm selected for GSoC, it will be my only job this summer.