Google Summer of Code

cdli

Organization Name: Cuneiform Digital Library Initiative (CDLI)

Transliterations editor and API

10th April 2020

Profile Information:

Name: Vishv Kakadiya

Email: krishnakakadiya9@gmail.com

Github: Vishv07

LinkedIn: https://www.linkedin.com/in/Vishv07

Location: Gujarat, India

Timezone: Indian Standard Time (UTC+5:30)

ABSTRACT

JTF is a new JSON-based format for transliterations that aims to make cuneiform textual data easily accessible for processing and modifications. It comes with a NodeJS API, jtf-lib, to provide an **ATF format** parser and converter, a CRUD interface, and a module for sign list operations, jtf-signlist. A React web application, uqnu, is being developed to import transliterations from files, validate, edit, export, etc. The task is to integrate this infrastructure into **CDLI's framework** and allow crowdsourcing and individual work on texts.

Mentor: Ilya Khait

PROJECT OBJECTIVES

- Framework integration:
  - The JTF API and web application run in a framework docker container.
  - The web application is accessible via a framework URL.
- CDLI database:
  - Stores JTF data.
  - Has a version control system that efficiently stores changes to transliterations.
- Framework API integration:
  - JTF API integration with the framework's public API.
  - JTF output function in the API.

IMPLEMENTATION OVERVIEW

1) Framework integration

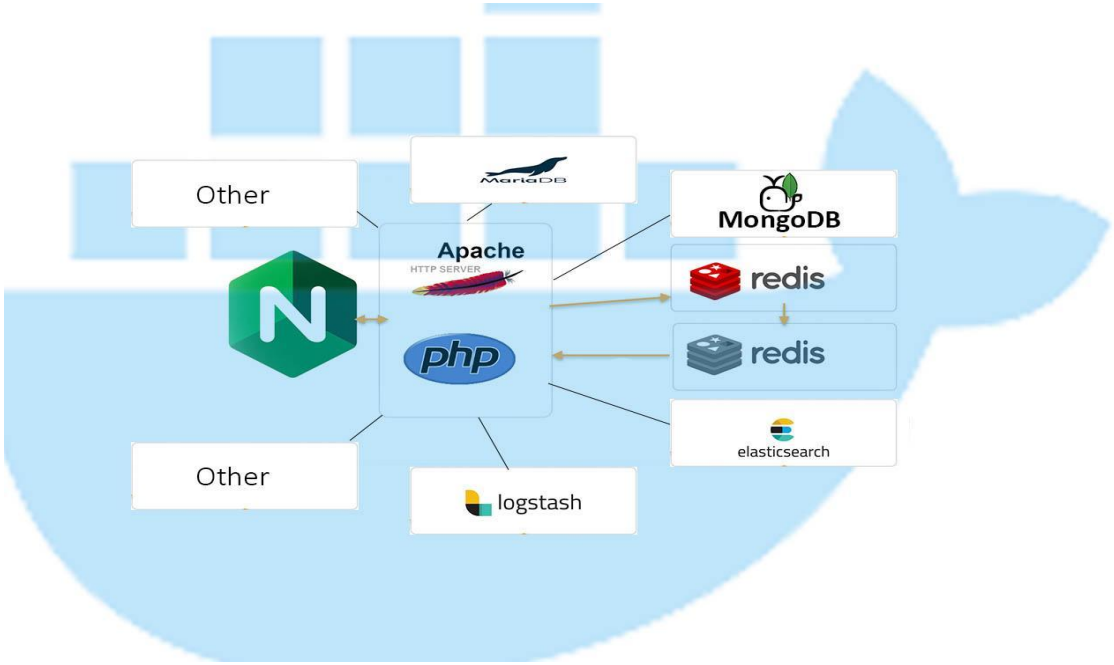
Integrating the **JTF API** and **uqnu** web app into the framework is the main and essential task for this project. Then only people will be able to use this Editor and can store their data.

Currently, Docker Container has various images which are running for CDLI's framework. So for the Integration part, we need this web app and API inside the running container. So It can be accessible through framework URL.

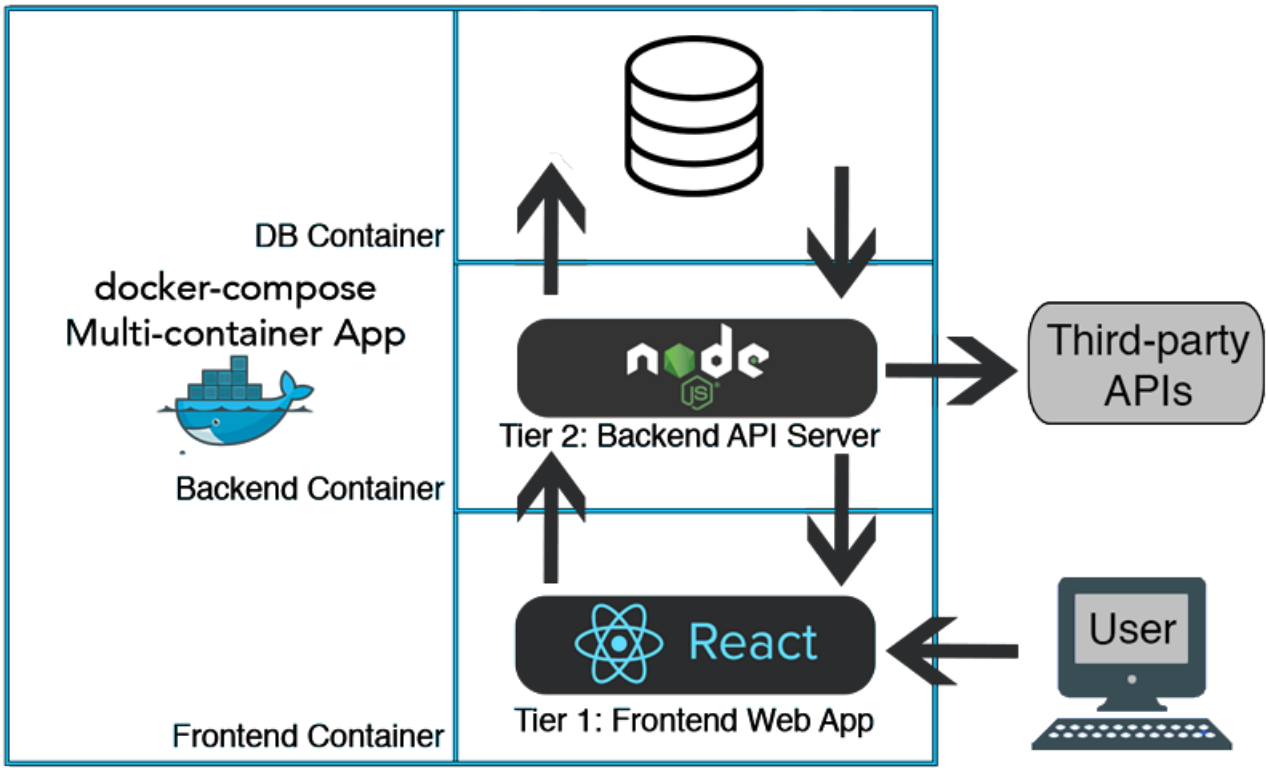
We can also specify this on the main web application's header for easy access for the users and we'll have our **uqnu** web app up and running on a specific port or default port's specific route.

for that, I'll add an image of a node from alpine which will be used for backend API and Frontend web app as well. and this separate container will also communicate to the database container to storing and retrieving the data.

Current Docker Container of framework: (Fig. 1)



Docker Container of this Project: (Fig. 2)



Above both Diagrams can clearly state that what needs to be done. Fig 2 will be the separate container for the whole web app and I'll integrate it in an existing container to up and running Editor web app.

2) CDLI database

Once the web app is integrated then the next task is to store and access **ATF** and **JTF** data. In **MariaDB** already **atf** field is there. I'll create one more field called **jtf** to store converted data. And later on, we can easily retrieve that data through a fetch query.

In order to store that converted data, I'll create a REST service or a query to store **jtf-data** for a particular author with a timestamp.

- **VERSION CONTROL SYSTEM OPTIONS.**
  1. **Git Server**
  2. **SVN**
  3. **CVS**

  I've had experience in setting up a **git** server for the version control system in Node JS. It is a great CLI tool that can easily solve this task. I will set up one **git** server with its **ACTIONS** so whenever any changes are applied to a text so It can store its version along with its data, author, and time.

  Git has slightly more advantage over the other two hence it is more powerful and yet most widely used so community support and its ability to utilizes multiple repositories and Multiple developers might work on the same project parallelly. So Git could be a good option.

  Surely there are other options like **SVN** and **CVS** for this task. As of now, I am considering git as my preference. If that won't seem to work then we can surely think about other options.

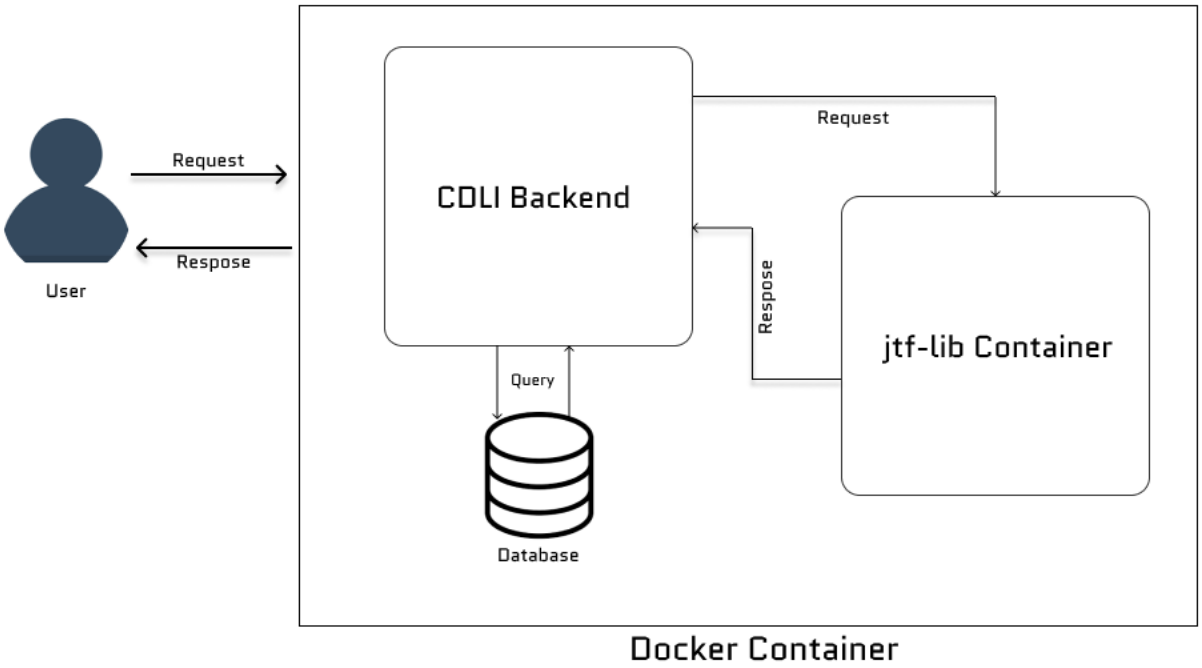
3) Framework API Integration.

I am considering **jtf-lib** will not require any changes for using its public functions and use it as a REST service so we can use it in our Framework's API so ideally, It will be accessible through Framework's API.

Hence Web app (editor & jtf-lib) will be integrated and it will be running in a Framework container before this task so we can communicate to it through the backend (**Cake php's Controller**).

I'll create a new endpoint (Route) in controller of CDLI's framework API. Which can send a request to jtf-lib (Inside a container) so jtf-lib will send a specific response which will be stored in **CDLI's Database.**

Workflow of Communication among docker containers: (Fig. 3)



- **Fallback Plan for jtf-lib:**

  Just in case jtf-lib needs some improvement I will allot a specific time to it and will make it more reliable. I am planning to add express as a framework to make HTTP requests and responses more reliable. Surely as of now, it doesn't seem that it requires anything. But if things won't go as I have planned so I have just mentioned them here.

- **Bonus Tasks**

  Well MERN stack is my strong domain where I have been working for the last 1 and half years. I've a couple of things in my mind apart from this listed task which I am willing to execute once Primary outcomes will get done.

For the GSOC, Outcomes will be my priority. But just in case I'll have more time than I'll try to get my hands on this task during GSOC. Otherwise, after GSOC, I will be active in the community and on this project.

CDLI Crowdsourcing functions:

- For this particular task many things have to be done first in order to implement the below features.
    - For that approval mechanism, we'll have to create a new service across the framework that can handle the request for adding or modifying transliterations data.
    - For Admins, we can implement authentication so they can use the web app. Hence the app would have already added to the framework so we can decide who can use it.
    - we will need a dashboard for this whole service and approval mechanism.
    - CDLI has an existing Dashboard either we can use that or we can make a new one.
    - I feel uqnu will need some improvement in order to run in the production for enabling this kind of service.
    - As mentioned in bonus tasks we can add more features like highlighting the changed data and some extra plugins for annotating linguistic features/images.
    - But before I get into this, core Outcomes will be my first priority and if time permits, then only I'll be getting into this part.

SCHEDULE

1. Community Bonding (May 17 – June 7, 2020)

- Getting acquainted with the organization, interacting with the mentors, and getting to know their desired way of working in the coming weeks.
- Learning more about advanced CakePHP concepts.
- Set up PR preview and git workflow in uqnu.

2. Week 1 (June 7 - June 13)

- Start working on the Integration part.
- Develop Container for a Web app.
- Develop Container for a jtf-lib.
- Writing tests for the web app.

3. Week 2 (June 14 - June 20)

- Continuing the work from the previous week.
- Integrate these two containers and check with manual testing.
- Make this web app accessible via a framework URL.
- Writing tests for the system.

4. Week 3 (June 21 - June 27)

- Continuing the work from the previous week.
- Start working on Storing JTF data.
- Making Endpoint (Route) to enable storing.
- Writing tests for the route and database.
- Writing a blog to update all the work done in the week.

5. Week 4 (June 27 – July 4)

- Continuing the work from the previous week.
- Setting up the Git server.
- Integrating the git actions which can eventually enable version control.
- Start researching and discussing the feasible approach with the mentor regarding this.
- Testing the version control system.

6. Week 5 (July 5 - July 11)

- Implementing the mechanism for the system that stores the data when changes occur to transliterations.
- Writing tests for the work done.
- Integrate it with the database and Cake Php so JTF data can be stored.

First Evaluation (July 12 - July 16)

- Manually **test** the work that has been done so far.
- Write a few **unit tests** for this and tests for services which have been made so far.

7. Week 6 (July 12 - July 18)

- Trying a different approach if the git server won't work as I have planned.
- Hence I have already mentioned a couple of options. Will give it a try.
- Writing a blog for the learnings in the week.

8. Week 7 (July 19 - July 25)

- Writing tests for work has been done so far for the CLI system.
- Integrating CLI System with the CDLI interface.

9. Week 8 (July 26 - Aug 1)

- Start working on JTF API integration with the framework's public API.
- Implement a Service which can communicate to both the docker container.
- Making a separate endpoint to store the JTF data in MariaDB.
- Writing Test for service which has been made.

Would be shifting to Fallback Plan for JTF-lib. Will spend some time on JTF-lib if it will require any changes or new features.

10. Week 9 (Aug 2 – Aug 8)

- Connecting Route to the jtf-lib container to get the response from it.
- Writing tests for the work done.
- Writing a blog for the work has been done so far.
- Working on Endpoint to store JTF data and make it accessible.

11. Week 10 (Aug 9 - Aug 15)

- Prepare the whole **Documentation** of the work I have done so far.
- Start working on bonus tasks, **if time permits.**
- Create a service layer in the framework that can get requests and enable admins to approve the request. So modification of transliterations data can be done.
- Updating all the work done in the blog.

Final Evaluation (August 16 - August 23)

- Manually **test** the project and finish details for final submission
- Updating all the work done in the blog.

DELIVERABLES

Weeks	Deliverables
Week 1 (June 7 - June 13)	• Makes the web app up and running inside a container.
Week 2 (June 14 - June 20)	• Enable service that communicates between newly added and already existed containers.
Week 3 (June 21 - June 27)	• Make routes(Endpoint) and controllers in CDLI backend to communicate with the service. • Store JTF data in the Database along with the author and timestamp.
Week 4 (June 27 – July 4)	• Integrating the git actions which can eventually enable version control.
Week 5 (July 5 - July 11)	• Implementing the mechanism for the system that stores the data when changes occur to transliterations.
First Evaluation (July 12 - July 16)	• Manually **test** the work that has been done so far. • Write a few **unit tests** for this and tests for services that have been made so far.
Week 6 (July 12 - July 18)	• Trying a different approach if the git server won't work as I have planned. • Writing a blog for the learnings in the week.
Week 7 (July 19 - July 25)	• Integrating Version Control System with the CDLI interface.
Week 8 (July 26 - Aug 1)	• Start working on JTF API integration with the framework's public API • Implement a Service that can communicate to both the docker container. • Making a route and method in the

	controller to store the JTF data in MariaDB.
Week 9 (Aug 2 – Aug 8)	• Working on a Controller to store JTF data and make it accessible.
Week 10 (Aug 9 - Aug 15)	• Prepare the whole **Documentation** of the work I have done so far. • Start working on bonus tasks, **if time permits.** • Create a service layer in the framework that can get requests and enable admins to approve the request. So modification of transliterations data can be done.
Final Evaluation (Aug 16 - Aug 23)	• Manually **test** the project and finish details for final submission • Updating all the work done in the blog.

TECH-STACK

- CakePHP
- Docker
- Git
- Bash
- NodeJS
- Webpack
- SVN / CVS (as a backup, if due to certain reasons, things not compatible with git)
- ReactJS
- Redux (State Management)

FUTURE WORK

The web app can be extended a lot in the future. If time permits, I would like to work on some areas of the project. Some of the possible extensions are:

- User-friendly User-Interface.
- Highlight the Changed word & Loader components.
- One Compare Section that can show the difference between Input and Output.
- Desktop app for the editor so It will be easy for the people who are working with cuneiform data
- Refactoring the uqnu codebase so other contributors can easily adopt it.

OTHER OPEN SOURCE CONTRIBUTIONS

1. https://github.com/cdli-gh/uqnu/pull/3 (**merged**)
2. https://github.com/kothariji/SyntaxMeets/pull/99 (**merged**)
3. https://github.com/kothariji/SyntaxMeets/pull/47 (**merged**)
4. https://github.com/kothariji/SyntaxMeets/pull/33 (**merged**)
5. https://github.com/SimplQ/simplQ-frontend/pull/388 (**merged**)
6. https://github.com/SimplQ/simplQ-frontend/pull/364 (**merged**)
7. https://github.com/SimplQ/simplQ-frontend/pull/348 (**merged**)
8. https://github.com/SimplQ/simplQ-frontend/pull/285 (**merged**)

WHY I AM BEST SUITED FOR THE PROJECT

I have had experience in creating scalable web applications in MERN stack and PHP. Especially I've had experience in creating an **SSML (Speech Synthesis markup language) Editor** that converts plain text into SSML text that is used in the TTS (Text-to-Speech) API of Google and Microsoft Azure.

I think I would be a great fit for this role because I'm a passionate learner and I adapt to new work environments very quickly. I can vouch that if I'm selected, I will give my utmost dedication towards creating a piece of code that is scalable, efficient, and easy to read.

The opportunity to work on this project would be very exciting and would help me showcase my coding abilities and my interest in the field of Web development. I plan to be a long-term contributor and would be highly interested to support the organization and solve the issues in the tool developed by me even after GSoC 2021 has ended.

EXPERIENCE

1) **Frontend Developer [Listnr Co.] [May 2020 Sep 2020]**
   - Developed a Front-end for web application using React, Redux
   - Built SSML Editor that helps to convert simple text to SSML text
   - Built End-to-End payment recurring service for customers to use this product.
   - Built a Complete SaaS Product along with the Developers

2) **Software Engineer Intern [Silvewing Technologies] [June 2018 May 2019]**
   - Learned the development process of building a software
   - Created stable back-end code for the requirements using PHP.
   - Built front-end for Progressive web apps and 2+ projects along with the senior developers.

PAST PROJECTS

1) HEALTH CITY [Python, Php, Mysql, Javascript, Numpy, Pandas]

- Developed full-stack web application that helps hospitals and governments track health-related data of users and carry out analytics.
- Implemented Data Analysis Of Diseases Spread (Area Based, City Based)
- Visualized disease outbreaks across regions in the state across its regions to help authorities.
- Generates Health-Card for every user.

2) LOREMGINZO [Php, Mysql, JAVASCRIPT]

- Developed a Complete E-commerce Website for a Client.
- Real-time E-commerce website to buy clothes.
- Implemented PayUmoney as a payment gateway.

3) COVID-19 TRACKER [React JS, Redux, REST API]

- Data Visualization for Covid-19 cases across the world
- Github Code: Covid Tracker
- https://react-js-co-vid-19-tracker.now.sh/

ABOUT ME

1. Contact Details

a. **Name:** Vishv Kakadiya

b. **Alternative Email Address:** auseonly34@gmail.com

c. **Telephone:** +91-9067740572

d. **Time zone:** Indian Standard Time (UTC+5:30)

2. Current Education Details

a. **College:** LD College of Engineering

b. **Degree:** B.Tech. In Information & Technology

c. **Current Year:** 3rd Year

d. **Expected to graduate:** May 2022

3. Achievements and Positions of Responsibilities

a. Selected for finals in Smart Gujarat Hackathon.

b. Selected under Student Startup and Innovation Policy, Gujarat.

c. Selected as one of the top 12 teams in the OPPO Fintech Contest at IIT-Bombay.

d. **Contributor at Girl script summer of Code (GSSOC):** I am currently an open-source contributor at GSSOC under the project **SyntaxMeets**. (My Contributions)

OTHER COMMITMENTS DURING SUMMER

I am not participating in any other internship during the GSOC program. In May-mid I won't be available for 5/6 days as my End Semester exams will be going on. But After that, I'll give 20 hours extra Just in case I need it. Except that I will be able to devote 35-40 hours per week and try for more if required.