GSoC 2018 - Project Proposal

Improve Demo creation in Origami

Mentors: Avais Pagarkar(@AvaisP) Utkarsh Gupta(@uttu357)

Basic Information:

Personal Info:

- **Name:** Deepesh Pathak
- **Nick:** fristonio, dpathak
- **Github**: fristonio
- **Email:** deepeshpathak09@gmail.com
- **Phone No.** +91-9456170426
- **Location:** India
- **Time Zone:** India (UTC+5.30)
- **Webpage:** fristonio.co
- **Resume:** http://fristonio.co/assets/Resume.pdf
- **Skype:** live:deepshpathak
- **Education:** **II Year** B Tech. ECE Indian Institute of Technology Roorkee

Contact and work timings:

- Reachable anytime through **email**, **gitter** or a well planned video session if required.
- **GPG Key ID**: C9C33F71
- **GPG Fingerprint**: 1C23 0547 369B 7464 D84F FCF1 D9AA 53BE C9C3 3F71
- **Timings**: UTC 5:30 AM to 8:30 PM (IST 11 AM to 2 AM)
- Typical Working hours:
  - **Weekdays**: UTC 10:30 AM to UTC 3:30 PM, UTC 4:30 PM to UTC 7:30 AM
  - **Weekends**: UTC 4:30 AM to UTC 8:30 AM, UTC 10:30 AM to UTC 3:30 PM, UTC 4:30 PM to UTC 7:30 AM

Statement of Motivation

1. **What is your motivation for participating in Google Summer of Code?**
   - My motivation for GSoC this year is getting started with open source community. I strongly feel that GSoC is a great program and platform for introducing organizations with prospective contributors which can help to improve and grow both the individual and the community.

2. **Why did you choose CloudCV**
   - *CloudCV* is a great platform to get started with open source with a real fascinating aim to bring cloud and machine learning together. The mentors here are helpful knowledgeable and responsive. The reason for choosing Cloud-CV for GSoC is because it provides a good opportunity to apply my learning on a practical scale.

3. **Why this project idea**
   - I chose this particular project because it is very well aligned with my interests and also correlated to what I have worked with in the past. Other than that this is my field of interest and therefore the natural inclination to this project.

4. **What are your expectations from us during and after successful completion of the program**
   - This project will provide me an opportunity to apply my learnings on a practical scale. I wish to learn more through this phase and contribute to this project to my fullest.
5. **What are you hoping to learn**
   - Origami is an interesting project with some great base tech stack, I am very excited to try my hands on it. Besides this, I feel that working with the mentors is going to refine my skills all through this period.

Experiences

Muzi

- Muzi is a web based music player developed by SDSLabs for IIT Roorkee. I was involved in developing the complete frontend of Muzi from scratch.
- Muzi is a single page application, the tech stack of which includes *HTML,* SCSS *ReactJS* and *Redux* framework. Muzi uses *sound.js* for its song manipulation capability.
- It involved building a large no. of modular components with a lot of user interactions and animations, managing authentication and interacting with an API, along with writing unit tests and end to end tests.
- Muzi was both the overall and web category winner of Software Development Section organized competition in *Srishti(*Annual intra tech festival of IIT Roorkee).

Howl

- Howl is the backend for muzi, it provides a REST API for interaction with the database. I was involved in the refactor of entire backend along with the addition new features to Muzi.
- Howl is build over the modern PHP web framework *Laravel*. I added many new features to Howl including songs likes, playlist stars and refactor of previously used authentication system with new central auth of SDSLabs,
- Along the lines of howl exist *muzi-scanner* which intelligently scans muzi's CDN and indexes tracks with metadata into the database, I was involved in its refactor too.

A.V.A.D.H.I

- *Absence Verification and Detection Helping Instrument* is a packaged application that helps keep track of attendance in an organization.
- It is a complete ecosystem consisting of a Laravel based REST API, a web application for user registration and attendance visualization, a mobile application in JAVA to track attendance using mobile *Fingerprint sensor,* a cross platform desktop app built over electron to log attendance along with a bot to interact with the REST API provided.

- The application was build over a period of 5 days. I worked on creating the desktop electron application, Android application and AVADHI network scripts in the project. The code for the application can be found at A.V.A.D.H.I
- A.V.A.D.H.I was the campus winner of online round of Code.fun.do and secured 4th position in the offline round of the same. It also won two 1st prizes in IIT Roorkee STC organized *Srishti event* in **Dynamic website** and **Network software** category.

Predictify:

- Predictify is a web based machine learning application which aims at predicting and giving user useful feedback according to academic data provided.
- It uses the dataset provided for prediction by *OpenEdAI* community. The application was built over the period of 7 days with components including a vanilla CSS/JS backed frontend, a django based REST API and python based machine learning model.

g-Ignore

- g-Ignore is a python package which automatically and intelligently generates gitignores for your git projects.
- Unlike other automated tools it does not unnecessarily bloat .gitignore, but only includes those files which are present in the repository to ignore along with option to replay your previous versions of gitignore.

Hackathons

- I regularly participate in hackathons and other competitions. I have developed numerous applications in such hackathons including A.V.A.D.H.I Predictify and RISC Processor

There were also a number of personal projects I have worked on, they can be found at my github profile at github.com/fristonio

CloudCV Contributions

I developed a passion for programming and software development in my freshman year, since then I have been actively exploring this field. I am an active open source contributor. I write clean code with exhaustive unit testing and concise documentation. I try to keep my code up to mark by giving importance to even the minor details like indentation, grammar and spellings. It's been just a month contributing to *CloudCV* but I have made some significant contribution to the community. Apart from regular contribution to the codebase I am also an active member in CloudCV gitter channel. I give regular input to Origami related discussions and issues. My previous involvements in *CloudCV* includes patches to **Origami** mentioned below.

Pull Requests

- #128(merged): Bump react-router to version 4, dealt with all the breaking changes it brought to the codebase.
- #140(merged): Improved the UI for create demo page making it responsive.
- #145(merged): Refactored package.json adding scripts for frontend production build, eslint and tests and fixed other broken scripts.
- #189(merged): Added frontend build script to travis removing redundant webpack-stats-local.json
- #187(merged): Fixed lint prettier lint errors in the codebase.
- #135(merged): Removed console errors related to inline style attributes usage.
- #142(merged): Fixed broken state of navbar links and its UI.
- #179(merged): Fixed link to help on initial setup page.
- #178(merged): Fixed redirect to home on origami icon click in sidebar.
- #199(merged): Added limit to demo description length in demo card.
- #156(merged): Fixed main page layout height and footer alignment.
- #159(merged): Fixed UI of search option selector to show complete option and made it responsive
- #160(merged): Toastr imports fixed in origami frontend components module.
- #192(merged): Included new setup instructions in readme and fix lint warnings

Issues

The best way contribute to project is not just by solving issues but by first finding them, working on the same lines I regularly use Origami to find new issues. A list of issues created by for Origami is mentioned below.

- #194(closed): **[UI]:** Limit description length to be shown in demo card.
- #193(open): **[Feature]:** Fix profile page of users to include more relevant information.
- #191(closed): **[Docs]:** Update readme to include new setup instructions.
- #190(open): **[Backend]:** Add restriction on demo name in create demo page.
- #177(open): **[Core]:** Fix docker build setup for Origami.
- #170(open): **[UI]:** Direct navigation to create demo page does not work.
- #169(open): **[Feature]:** Origami Insights.
- #163(open): **[Backend]:** Fix duplication of cover-image for demo in database.
- #162(closed): **[UI]:** Fix origami icon click on sidebar.
- #158(closed): **[Style]:** Fix toastr imports in origami frontend components.
- #157(closed): **[Frontend-Core]:** Remove *webpack-local-stats.json* from repository.
- #155(closed): **[UI]:** Fix footer render state for no user logged in state.
- #150(open): **[UI]:** Redesign initial setup page.
- #150(open): **[UI]:** Fix background color and demo card grid layout in MyDemos page.
- #149(closed): **[UI]:** Fix search option selector width to show complete options.
- #144(closed): **[Core]:** Introduce prettier and eslint to Origami.
- #143(open): **[UI]:** Make demo page UI responsive and user friendly.
- #141(closed): **[UI/UX]:** Sidebar does not show correct active links when we try to open a url directly.
- #139(closed): **[UI]:** Improve create demo page UI and make it responsive.
- #138(closed): **[UI]:** Fix help button on inital setup page.
- #132(closed): **[Core]:** This issue is a parent issue which tracks the progress of following
  - Move prettier and eslint-plugin-prettier to dev dependencies.
  - npm lint scripts do not work at all, the directory provided is not correct.
  - Fix npm scripts to run origami in dev environment.
  - Fix test script and add dummy test for origami as of now.
- #129(closed): **[UI]:** Make main layout responsive and remove unnecessary scroll.
- #126(open): **[Frontend]:** Fix AJAX network calls in frontend react components.
- #127(closed): **[Frontend]:** Fix react inline styling in react components.

Open Source Contributions

Other than CloudCV I have also contributed to various other open source projects. I am regular contributor to some big open source project like **The Tor Project, OWASP** and **MITMProxy** my contributions to these organizations can be viewed below

- theTorproject/Tor-core - https://github.com/torproject/tor/commits?author=fristonio
- OWASP/owtf - https://github.com/owtf/owtf/pulls/fristonio
- mitmproxy/mitmproxy - https://github.com/mitmproxy/mitmproxy/pulls/fristonio

For other contributions view my github profile fristonio

Project Details

Cloud-CV is an open source cloud platform with the aim to make AI research reproducible. Origami (previously called CloudCV-fy your code) is an AI-as-a-service solution that allows researchers to easily convert their deep learning models into an online service that is widely accessible to everyone without the need to setup the infrastructure, resolve the dependencies, and build a web service around the deep learning model. The current codebase of Origami is not up to mark and is still evolving, it needs some refactoring in the area of *demo creation* which is essentially the heart of Origami. I intend to work on the idea of improving Demo Creation in Origami which includes solving existing issues in demo creation, proposing and implementing some new features and reporting new issues that exists.

The current implementation of Demo Creation in Origami is not ideal and is still evolving, there are a lot of deadlocks where it struggles and is not very intuitive. Providing user a way to easily create and manage demos for their deep learning model is the primary goal of *Origami* which can be greatly improved. To enhance user experience some changes are needed in UI and in feature list of *Origami.* Incorporating all these changes would be the primary goal for my GSoC project.

Project Tasks

1. Analytics page for your demos(Origami Insights)

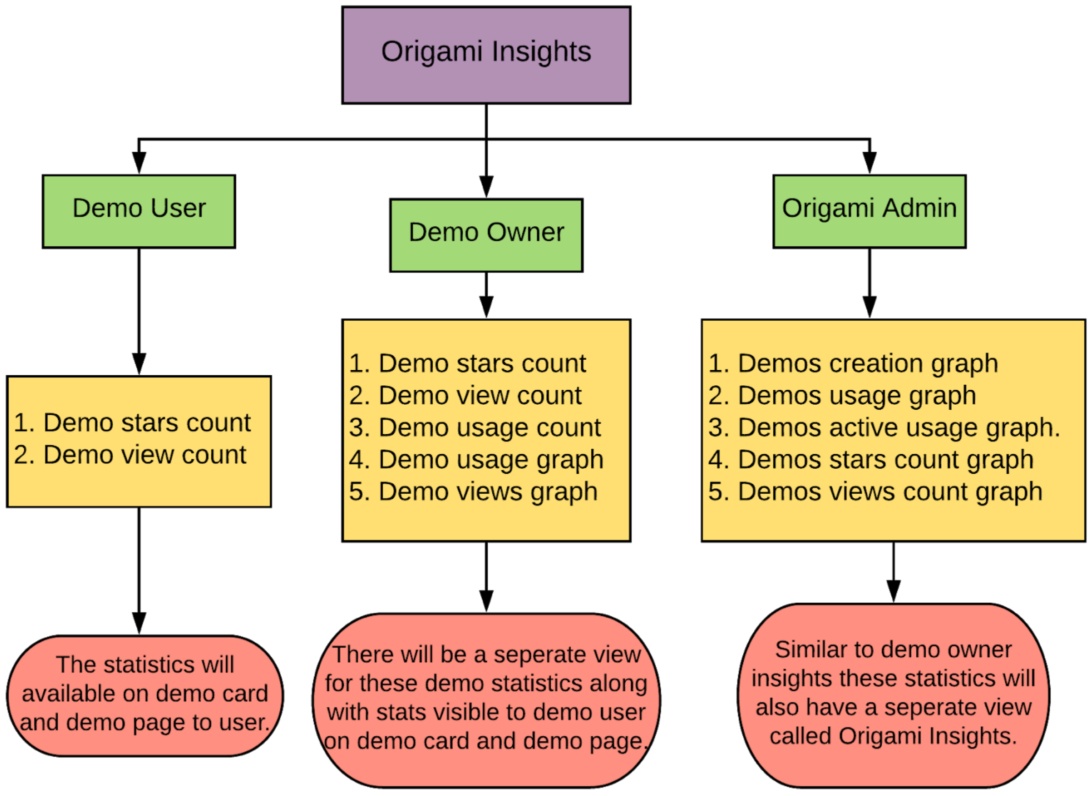
1.1 Overview

- Origami as of now does not provide user any statistics about the demos. This is a must have feature for origami to give user insights about a demo.
- I have already started working on the idea and there exists an issue created by me that tracks the progress of this here
- Some of my initial ideas for the same include
  - Demo stars
  - Demo views
  - Demo usage count
  - Graphical analysis of demo usage

1.2 Flow

- For a demo Origami users can be of three types *Demo User, Demo owner and Origami admin.* The insights for each category will differ from the other.
- **Demo User**
  - A simple demo user will have access to only simple statistics about the demo, the only stats accessible to this user will be
    - *Demo Stars*
    - *Demo Views*
  - There will be no separate views for these statistics rather they will be shown in the demo card and demo page itself.
- **Demo Owner**
  - Demo owner will have access to all the insights related to the demo. So for each demo the owner will have a seperate view showing these statistics. Just like a normal user owner will also be shown simple stats like stars and views in demo card and page itself. The view on the other hand will list all the statistics.
  - For views of a demo I propose we show a graph showing views for that demo per day to the demo owner.
  - An extra feature for demo owner is a graph showing number of demo usages per day.
- **Origami Admin**
  - Admin will have access to only high level information related to the demos rather than having statistics about each demo separately. There will be a seperate page for showing statistics to admin.

- The statistics page will show the following details
    - A graph showing number of demos *created* each day.
    - A graph showing number of demos *used* each day.
    - A graph showing number of *starred* on demos each day.
    - A graph showing number of demos viewed each day.


Origami Insights
Demo User
Demo Owner
Origami Admin
1. Demo stars count
2. Demo view count
1. Demo stars count
2. Demo view count
3. Demo usage count
4. Demo usage graph
5. Demo views graph
1. Demos creation graph
2. Demos usage graph
3. Demos active usage graph.
4. Demos stars count graph
5. Demos views count graph
The statistics will available on demo card and demo page to user.
There will be a seperate view for these demo statistics along with stats visible to demo user on demo card and demo page.
Similar to demo owner insights these statistics will also have a seperate view called Origami Insights.

Origami Insights - Structural view

1.3 Implementation

➢ The flow I have proposed above will require the creation of some new Models for the database along with changes to schema of some pre existing models.

- **Model DemoStars -** New model containing starred demos with a mapping between demo and user who starred the demo.

id(Primary Key)
demo_id(Foreign Key)
user_id(Foreign Key)
updated_at(Timestamp)

- **Model Logs -** New model containing logs, each row will correspond to usage of a demo by user. The model will have the following attributes.

id(Primary Key)
demo_id(Foreign Key)
user_id(Foreign Key)
timestamp(Timestamp)

- **Model Demo -** Add new attribute to this model storing views for a particular demo. I do not propose to create a new model for this and then calculate views from that model because in that case the large number of entries will become highly resource intensive.

views(Integer)

- Once the models are set up API endpoints would be created to get the data required
    - For stars and views on a demo two extra fields will be added to demo info, views will be automatically used from demo model and we will get stars by annotating demo info query with Count of demo stars from DemoStars model.
    - For example

```
Demo.objects.filter(user_id=id).annotate(stars=Count('demostars'))
```

    - Model **Logs** will be used to get the number of demo usages, which will be available through an API endpoint.
    - To provide data for Origami Admin analytics API endpoints will be created using pre existing models.
    - Analytics views for demo owner and Origami admins will be implemented by a react component which will use charting library rechart to plot the graphs with the data provided by the API endpoint.

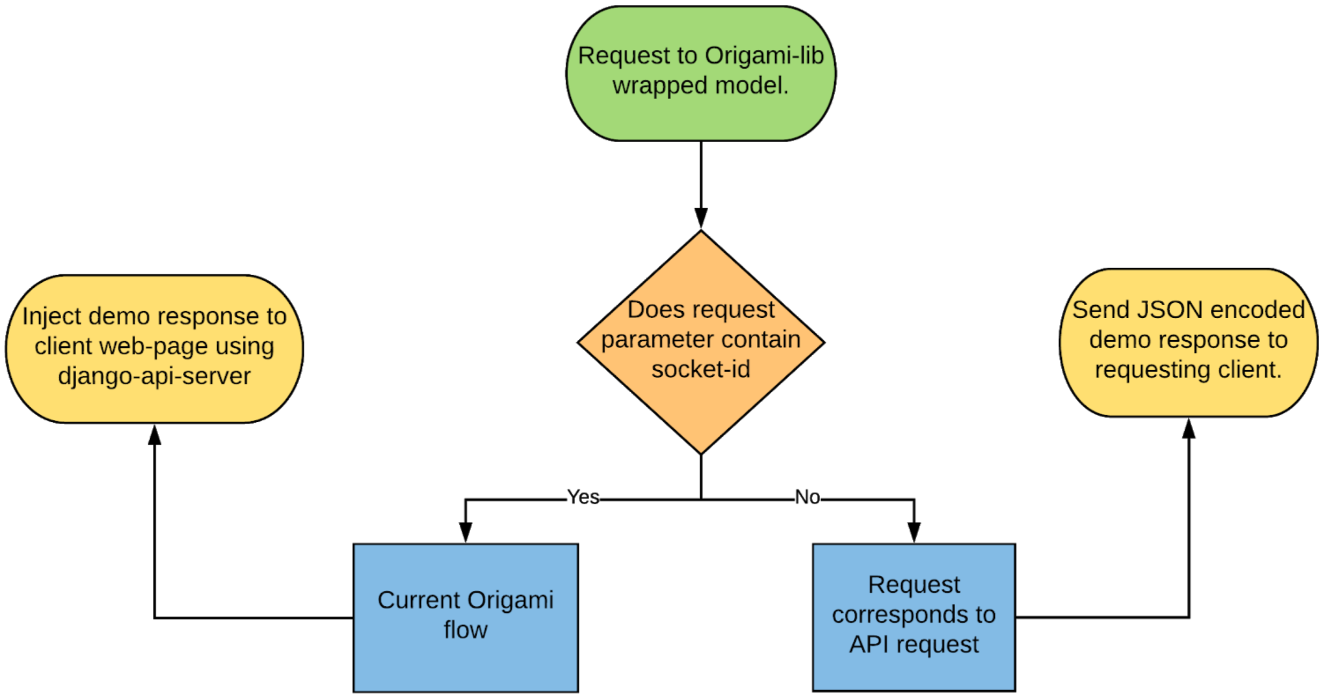
2. Provide REST API to access models

2.1 Overview

- ➢ Demo created by a user using origami-lib can only be accessed using origami website, providing user access to their models using REST API will give user more control over the application.
- ➢ Using REST API users can add their own customizations to demo output, this would make the core concept of origami *AI as a Service* much more flexible.

2.2 Flow

- ➢ Currently Origami only provide one method to interact with the model, the typical flow to provide REST API interface would be


Request to Origami-lib wrapped model.
Does request parameter contain socket-id
Yes
No
Current Origami flow
Request corresponds to API request
Inject demo response to client web-page using django-api-server
Send JSON encoded demo response to requesting client.

REST API access to demo models flow.

2.3 Implementation

- ➢ The current implementation of **Origami-lib** responds to a connection request by sending a POST request to *django-api-server* when the field socket-id is present in request form parameters and raises an exception otherwise.
- ➢ To provide API interface to access models we need to make changes to *origami-lib* such that whenever no *socket-id* is specified with the request the library should assume that the response should be an API response.
- ➢ Thus rather than raising an exception in **validate_socket_id** when no socket-id is found the function should return false, which will be used further to identify the request as an API request.
- ➢ Once it's established that the request is expecting an API response, the flask webapp running will send **JSON encoded response** data directly to the client without interacting with the *django-api-server.*
- ➢ *So essentially origami-lib will send the same response injected by django-api as JSON response when request is sent for API.*

3. Configure auto deployment and deploy Origami

3.1 Overview

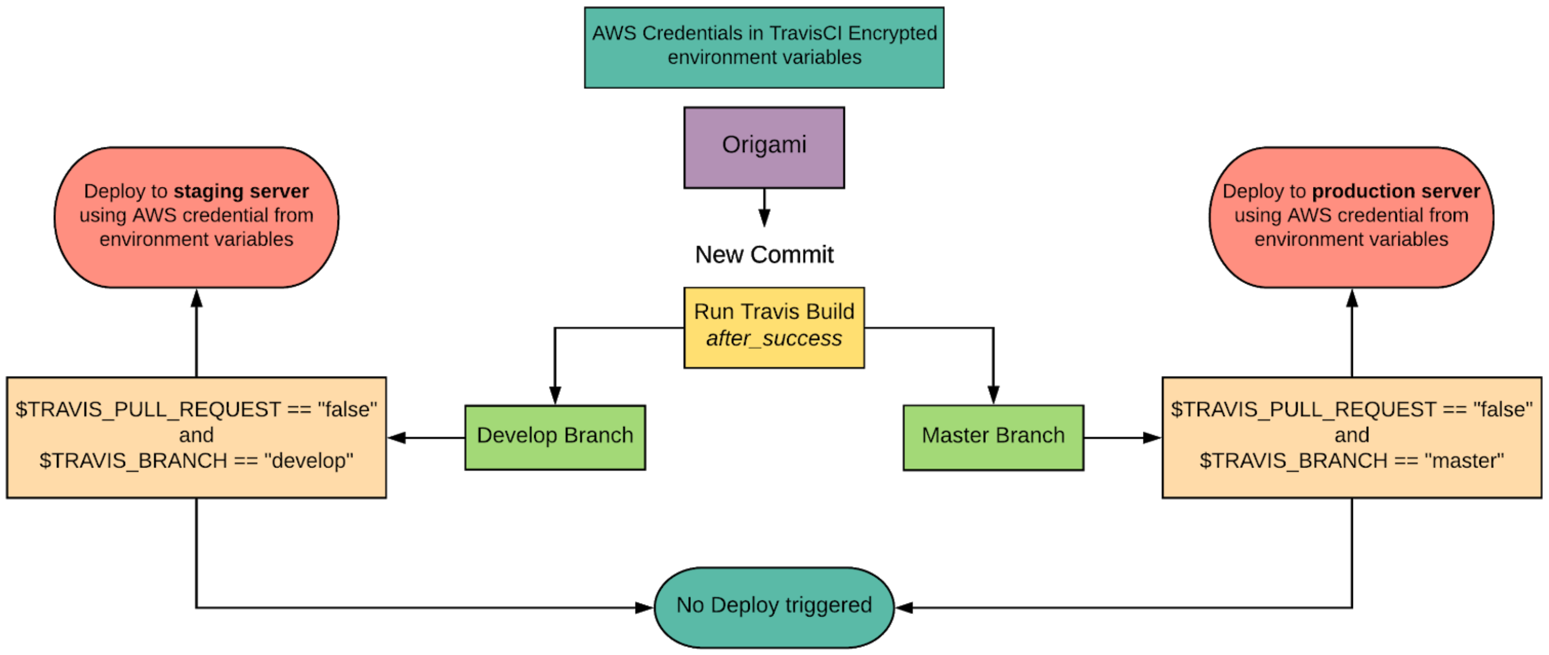
- Origami does not have any auto deployment mechanism as of now, so whenever a commit is made to master it has to be deployed manually on the server. This task can be easily automated by setting up **CI/CD** pipeline for Origami.
- This will reduce the manual input of server admin to deploy and test each new commit on the project.

3.2 Flow

- The project repository will have two branches
  - **deploy -** It will comprise the latest changes to repository by pull request.
  - **master -** It will act the main branch which will be actually deployed as production build.
- **deploy** will be merged into master once a *milestone* of Origami have been reached, and all the pull requests will correspond to a merge in deploy branch.
- The flow for CI/CD pipeline will use **TravisCI** build process.So once a commit is added to any of the two branch TravisCI will trigger the Travis build, this build on successful test pass will trigger a script which will deploy the latest commit on a staging server or main server based on the branch on which build was triggered.

3.3 Implementation

- The server login credentials which will be used to run deploy script on server will be stored in **TravisCI** encrypted environment variables and thus will be safe.
- Once a Pull Request is merged to deploy branch it will trigger a travis build which on successful completion will run the script to deploy the changes to staging server. This script will assert if the build is triggered by a pull request and is triggered on *deploy* branch. Once the mentioned conditions are asserted the latest commit will be deployed to staging server in a docker container.
- Similar flow will take place for *master* branch where the changes will be deployed on the production server.
- Currently *docker-compose* does not work properly and it need some refactoring, there is an outstanding issue for the same, resolving which should be the first step for the task.


AWS Credentials in TravisCI Encrypted environment variables
Origami
New Commit
Run Travis Build after_success
Develop Branch
Master Branch
$TRAVIS_PULL_REQUEST == "false" and $TRAVIS_BRANCH == "develop"
$TRAVIS_PULL_REQUEST == "false" and $TRAVIS_BRANCH == "master"
Deploy to staging server using AWS credential from environment variables
Deploy to production server using AWS credential from environment variables
No Deploy triggered

Origami - CI/CD pipeline flow

4. Drag and Drop demo creation

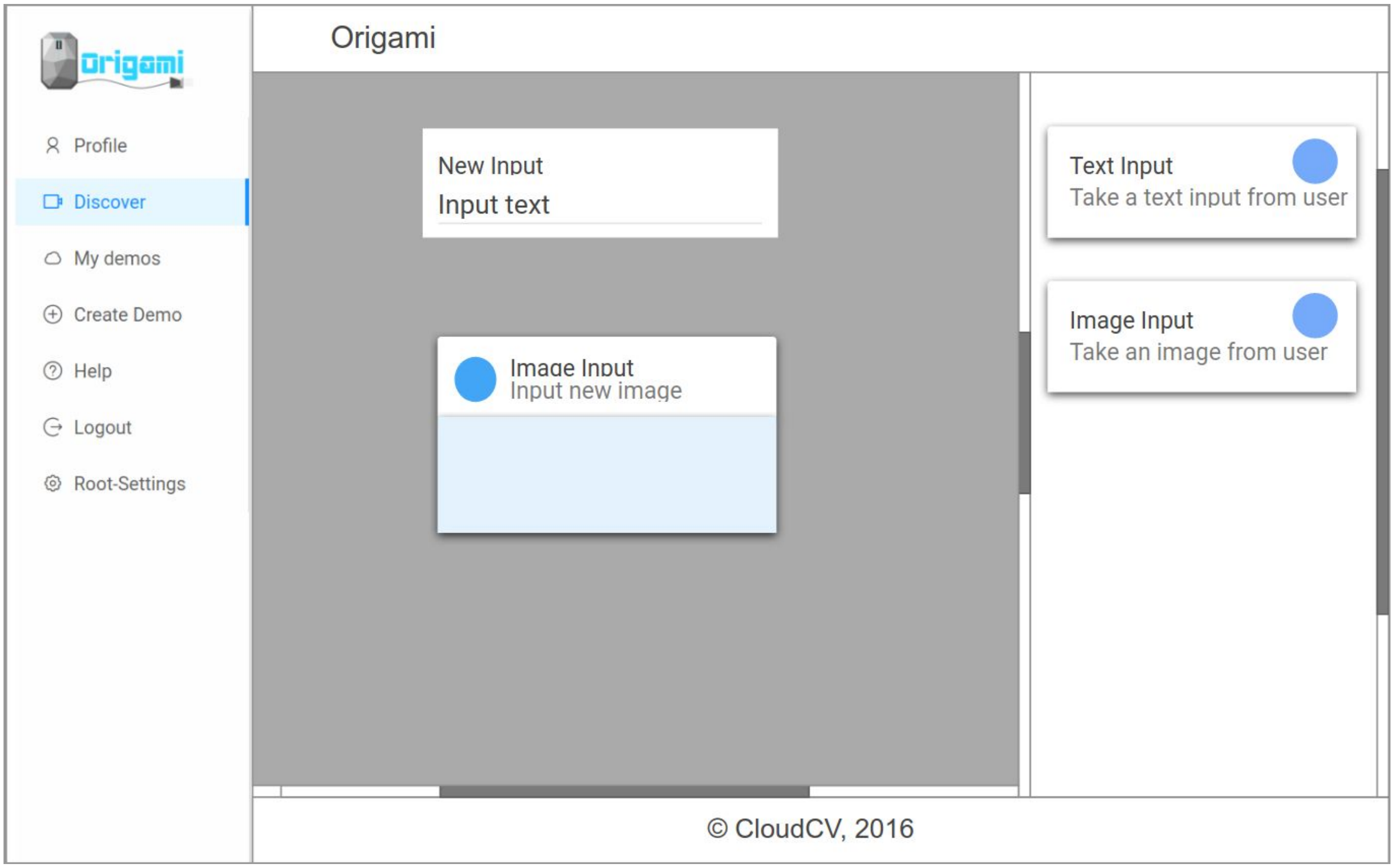
4.1 Overview

- ➢ The current method of specifying input and output component is neither very intuitive nor it is flexible. As of now user has to manually save one type of selected component first and then again navigate to same page to select a different type on component.
- ➢ Drag and Drop feature to select input and output component will enable user to easily manage and modify these components.
- ➢ The end result for the idea will be a functionality to allow user to select from given type of components and create a list of components using drag and drop from this list.

4.2 Flow

- ➢ Drag and Drop feature is a UI/UX enhancement and will be implemented in two pages
    - ○ Input Component selection page
    - ○ Output Component selection page

➢ A mockup for the same is attached below



Drag and Drop demo creation Input page mockup.

➢ The same flow will be associated with output component selection page.

➢ The user will be allowed to create a map of components using drag and drop.The drag and drop container will itself act as a preview for the components map so once a component is dropped it will be shown as a preview.

➢ On hover a **cross(x)** icon will appear over the component preview which can be used by user to remove component from the map. A single button click will the save the entire map created into the database and will direct user to next step.

4.3 Implementation

- ➢ React provides easy to use flexible libraries to implement drag and drop feature, I will be using react-sortable-hoc library due to its great support and flexibility. Also I have previously worked with the library so the task won't be difficult.
- ➢ Going according to the mock design I have proposed I will be needing to create two new components:
    - ○ A **sidebar** containing a list of components to select from for drag and drop. The components in this sidebar will be based on *material-ui* design followed throughout the repository. Each component in this container will be a draggable component which can be placed in the components map container.
    - ○ A component for **Drag and Drop sink** which will also act as a previewer for the current components map. This will be a fixed height container with scroll in both x and y direction.
- ➢ A single save button will be present which will allow user to save the current configuration of components map. When clicked it will trigger the *inputcomponent and outputcomponent* endpoints of API to save the component map state into database.
- ➢ As an additional feature the components map container will contain a full screen mode wherein the height and width of this component will expand to *100vh and 100vw* respectively showing the entire preview.
- ➢ One good idea for drag and drop component will be store the state of the current component map to the localstorage, so user can retrieve back, the created map if the tab is closed unknowingly.
- ➢ To implement this I propose to add a *localstorage* middleware to these component which will retrieve the previously created map if any from browsers local storage.

5. Add support for dialogue like interface to Origami

5.1 Overview

- ➢ Origami as of now only allows user to specify some input components and using these components and the machine learning model it provides some output information which is injected to output components.

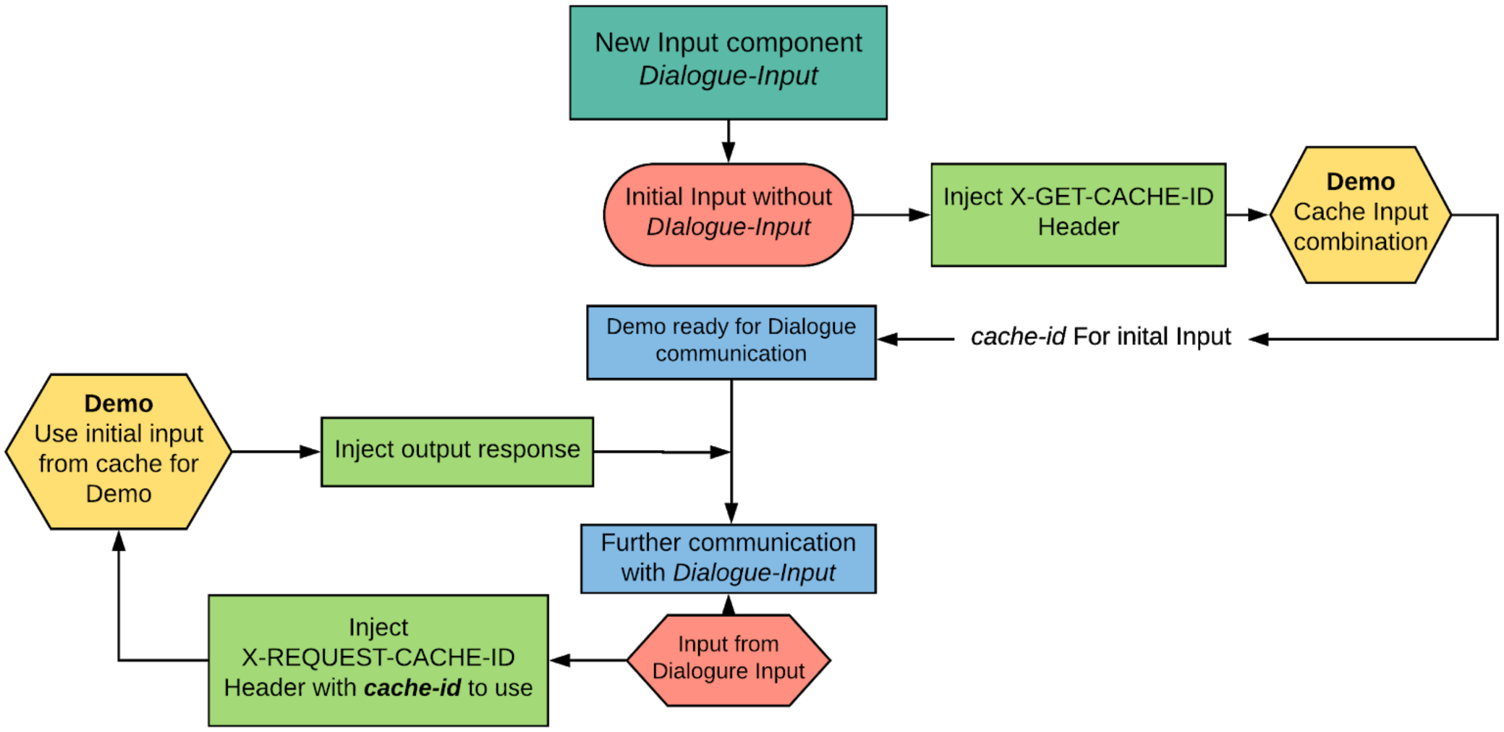
- ➢ To expand Origami one of the must have feature is the ability for a user to communicate back and forth with the demo. This will give origami ability to host demos like VQA.

5.2 Flow

- ➢ To enable user to communicate back and forth with origami-lib wrapped model demo we need to make some changes to origami-lib which include checking if user is interacting with dialogue interface and caching initial inputs if he is.
- ➢ Further communication with the model will be done using **Dialogue-Input** which will use the cached initial inputs for interacting with the model.

5.3 Implementation

- ➢ A new input component of type **Dialogue-Input** has to be added to Origami frontend to provide interaction using dialogues for back and forth communication with the demo.
- ➢ Origami can cache the image and text provided by user, each cached request has a unique ID(a MD5 hash). To provide a dialogue like interface to user we can use the same initial input from cache rather than making user send them again and again.
- ➢ To implement this user will be provided with the *cache-id* based on his initial inputs on his first interaction with the demo which will be send with the header **X-GET-CACHE-ID** set. This will send a response header **X-REQUEST-CACHE-ID** with the cache-id as a value.
- ➢ After that inputs from **Dialogue-Input** will request the demo with this header. Demo on encounter of this header will use *inputs from cache* to bootstrap the demo model and input from user will be used for interaction.
- ➢ Typical flow will look something like -


New Input component *Dialogue-Input*
Initial Input without *DIalogue-Input*
Inject X-GET-CACHE-ID Header
Demo Cache Input combination
cache-id For inital Input
Demo ready for Dialogue communication
Demo Use initial input from cache for Demo
Inject output response
Further communication with *Dialogue-Input*
Inject X-REQUEST-CACHE-ID Header with ***cache-id*** to use
Input from Dialogure Input

Flow of dialogue interface architecture for origami

6. Identify and fix all UI bugs along with Origami code refactor

6.1 Overview

- ➢ The codebase of Origami is still evolving and there are a lot of UI issues that exist. To make origami more usable and user friendly the UI must be intuitive and responsive.
- ➢ Finding and solving UI issues and enhancing user experience in origami is the primary goal of this idea.
- ➢ This a long running process and will be implemented in parallel to all other tasks throughout the coding phase of GSoC.
- ➢ I already had some involvements in this task before GSoC. I added prettier and eslint support to Origami along with lint scripts to *package.json*.

6.2 Flow

- ➢ This task will require using Origami as a user to find UI/UX issues and reporting them to github issue tracker.
- ➢ Fixing all issues tagged as ***UI*** and ***Frontend*** will be among the primary deliverables of this Idea.
- ➢ Some other issues that needs to be addressed in this task will include
  - Bootstrapping most of the existing frontend code blocks into components to make it more usable.
    - One example for where this issue exist is *Demo card* in various components like *User Profile Page, Discover Page and User Demos Page.* The current implementation have Demo Card code hardcoded to all three pages.
    - This is redundant behaviour and can be resolved by making a single separate component for DemoCard and then importing it in each page.
  - Add guidelines to create new UI components in origami.
    - This will also include first refactoring frontend code to a more flexible, usable and readable implementation.

6.3 Implementation

- ➢ The implementation of the idea will deal with resolving and finding UI/UX bugs. The two tasks will go in parallel.
- ➢ I have already filed some issues related to UI/UX which can be found here and will report more of them as I will progress implementing other tasks.
- ➢ Some of the key points I will keep in mind while looking for UI bugs in Origami are
  - UI should follow material design.
  - UI must be intuitive for user.
  - UI must be responsive.
  - UI code must be reusable.

7. Writing robust unit tests for Origami-lib and Origami repository.

7.1 Overview

- ➢ Origami has a good test coverage for the django API but still there are some improvements that can be made for better testing. Origami-lib test coverage on the other hand can be increased.
- ➢ As of now no frontend tests exist in Origami, adding tests to frontend will make entire project more robust and less prone to erroneous behaviour.
- ➢ The primary deliverables of the idea will be to add frontend tests and improve the coverage of backend tests.

7.2 Flow

- ➢ This a long running process and thus will be followed throughout the coding phase, each code contribution I will make to the codebase will be followed by its own test suits.
- ➢ **JEST** is a powerful and flexible javascript testing framework, integrating it with the current react code will be the primary deliverable for this task.
- ➢ I have chosen JEST for frontend testing because of the easiness it provides in creating all kind of mocks that is essential when it comes to unit testing.

7.3 Implementation

- ➢ The implementation of the task will deal with increasing the test coverage of both Origami and Origami-lib codebase and adding tests to Origami frontend.
- ➢ Initially to add frontend tests the javascript code must be bootstrapped with **JEST** testing framework.
- ➢ Adding frontend tests will initially focus on the following given parts
    - ○ Tests for redux store and reducers.
    - ○ Frontend test for create demo page.
    - ○ Frontend tests for Demo page.
- ➢ Adding some basic *snapshot tests* to Demo page and Demo creation page will give a basic foundation to better frontend tests which can be further extended later.
- ➢ Frontend testing can later be extended to entire frontend by testing using headless browser after GSoC.

TIMELINE

April 23 - May 14	Getting familiar with Origami codebase and team. Work on existing created issues in Origami repository. Read origami docs and other resources for implementation of GSoC tasks. Refactor *frontend UI* and code style of Origami. Make existing frontend components reusable by splitting existing components into more modular and flexible structure.
May 14 - May 27	Start working on Origami insights feature with the specifications mentioned above. Try to find more ideas which can be a part of analytics and try to possibly integrate them too. Work on increasing test coverage of django API
May 28 - June 8	Finish Origami insights and start working on the feature to provide REST API access to models discussing the changes to origami-lib with the mentor. Start working on adding frontend test to Origami. Bootstrap Origami frontend with JEST testing framework.
June 9 - June 15	Finish REST API access for demos feature along with tests after which user apart from using Origami to access demo can also make use of REST API. Prepare for **Phase 1 Evaluation.**
June 16 - June 29	Add CI/CD pipeline to autodeploy Origami to staging server and production server. Add basic tests for Origami frontend using JEST testing framework which will include tests for store and reducers.
June 30 - July 8	Add demo input and output component selection using drag and drop feature and start working out on outstanding UI issues in Origami repository. Report more UI issues and solve them following the new style guide for Origami with extensive frontend component reusability.
July 9 - July 14	Bootstrap all the works done till now with robust test suits, identify

	and fix UI bugs that might have been introduced in the earlier stages and prepare for **Phase 2 Evaluation.**
July 15 - July 22	Start working on support for dialogue like interface with Origami demos. Fix any existing issues in *origami-lib* and try to find more of them. Properly document *origami-lib* code with *sphinx* style documentation and add makefile to auto generate documentation.
July 23 - July 29	Complete support of dialogue like interface for Origami demos and start working on more UI related issues in Origami.e Refactor existing frontend API into a more modular and flexible format. Fix issues like #201 to improve code quality of Origami.
July 30 - August 10	Wrap up code by adding remaining documentation and add more tests for Origami codebase. Analyze and refactor all the previous code changes and possibly make a new release of **Origami.**
August 10 - August 14	**Preparation for final evaluation**

Secondary Goals

Given the time constraints of GSoC it is difficult to implement everything I have in mind for improvement of Origami. I have added some of those feature and improvements which I intend to add to origami in a list described below. These will be my secondary goals for GSoC and will be implemented if I finish the GSoC deliverables early. If not implemented during GSoC they will be worked on in the post GSoC period since I intend to work with the community even after end of GSoC coding phase.

1. **Add more Demos to Origami.**
    a. Origami lacks in terms of no of public demos as of now, one of my major post GSoC goal would be to add more demos to Origami in a dockerized format for easy deployment. Some of my initial ideas regarding the same are
        i. Image captioning demo
        ii. Demo for VQA

2. **Markdown based detailed demo description**
    a. Currently Origami does not give user much flexibility for writing a proper demo description. For now there only exist a simple input box asking for demo description while creating a demo.
    b. Providing user an interface to write a detailed description for demo is a much needed feature for origami.
    c. I aim of adding a markdown based real time editor for Origami to add a detailed description to demo. It will act as a feature to give user a deeper insight of the demo.
    d. In addition to markdown based editor a file upload utility will also be provided so that user can directly import his description from a markdown based document.

3. **Improving profile page**
    a. User profile page in origami is not at all intuitive nor does it shows relevant information regarding the user. It does need some refactoring.
    b. We can use github API to fetch user details and can use it in users profile page. After the implementation of *Origami Insights* feature we can also show statistics of user activity and demos on profile page.
    c. I have created an issue on github regarding the same and I intend to work on the same in post GSoC period

4. **Adding demo feedback support in origami**
    a. Currently origami does not provide a user any interface to give feedback regarding the demo to the Demo owner. This is a must have feature which will enable user to improve their demos.
    b. This will also increase reachability of Origami. The idea will deal with providing user a ***comment modal*** to communicate with demo owner. Rather than making it a one to one feedback I propose to associate a comment thread with each model moderated by the demo owner.
    c. For this we will need to create a new model ***comments*** in the backend which will have one to many mapping from demo to comments. Each comment will be associated with a parent attribute to give comment interface a proper thread like architecture.
    d. For the modal we can use easy to implement react-portals due to its highly flexible user interaction capabilities.

5. **Refactor existing codebase of Origami**
    a. Existing codebase of Origami could use some refactoring in the area of frontend request api which can be made more modular and flexible.
    b. Components can be made more react style by addressing console warnings. An existing issue for the same is mentioned here.

Time Commitment

1. **Do you have any other engagements during this period?**
   - No, I have no long term commitments during this period. I seldom get tired of coding and reading related stuff. I carry out these activities even for recreation.
2. **Do you have exams or classes which overlap with this period?**
   - There will be no clashes between May 13, 2018 to July 12, 2018. After July 12, however, I'll be having classes so I may be able to devote around 5 hours on weekdays and 8-9 hours on weekends.
3. **Do you have any other short term commitments during this period?**
   - I'll be on a family vacation from 3 July to 6 July, so won't be able to work during that time.

Coding Skills

Programming Languages and Frameworks

- Sound knowledge of OOPs and MVC Architecture and Docker environment.
- Proficient in **Python**, Haskell, PHP, C/C++, HTML/CSS, **Javascript**
- Moderate - JAVA, GoLang
- Frameworks - **React**, **Redux**, **Django**, **Flask**, Laravel
- Databases - MySQL and **PostgreSQL**

Development Environment

- Ubuntu or Debian 9
- Sublime Text and Vim along with git for version control

Other Interests

- Information Security (Web Exploitation, Rev-crypt and BInary Exploitation) and Open Source

Personal Background

I am a 20 year old, second year student currently enrolled in Electronics and Communication Engineering (IV Year Course) at IIT Roorkee. I developed a passion for programming and web development in my freshman year. I have been contributing to open source regularly since about four months from now - looking over to repositories of the products I use / come across and trying to give my part of contribution back to the organization whose product has been an asset to me.
I am an active member of SDSLabs at IIT Roorkee, a bunch of passionate enthusiasts trying to foster software development culture in the campus. I have a previous experience to work as per my proposed timeline with a mentor on a summer project in SDSLabs. Also most of our work is done according to a strict timeline with a lot of focus on test driven development and following best practices to structure code as per the design principles for the task.

References

- CloudCV GSoC website
- CloudCV GSoC Template
- TravisCI docs - Customizing the build
- Origami - Read the Docs
- react-sortable-hoc Library for drag and drop