X-Admin

Basic Information

Name : Harshil Jani
Majors : Electronics and Communications
Minors : Computer Science Engineering
Year : Second Year (Sophomore)
Institute : National Institute of Technology - NIT (Surat)
First Language : English
Github : Harshil-Jani
Email ID : harshiljani2002@gmail.com
Phone No. : (+91) 7984677391
TimeZone : Indian Standard Time (UTC +5:30)

Project

Abstract

Each government entity needs access to a database of relevant users and beneficiaries. The Education department needs to engage with students and teachers, The Agriculture department would engage with farmers and so on and so forth. So, it is important to store and manage beneficiary data with privacy and security safeguards in place. X-Admin is a low-code framework to create internal admin tools for governance use cases. This project is aimed at creating an auto generated registry management tool i.e. X-Admin for Sunbird RC. Some Features to be implemented are

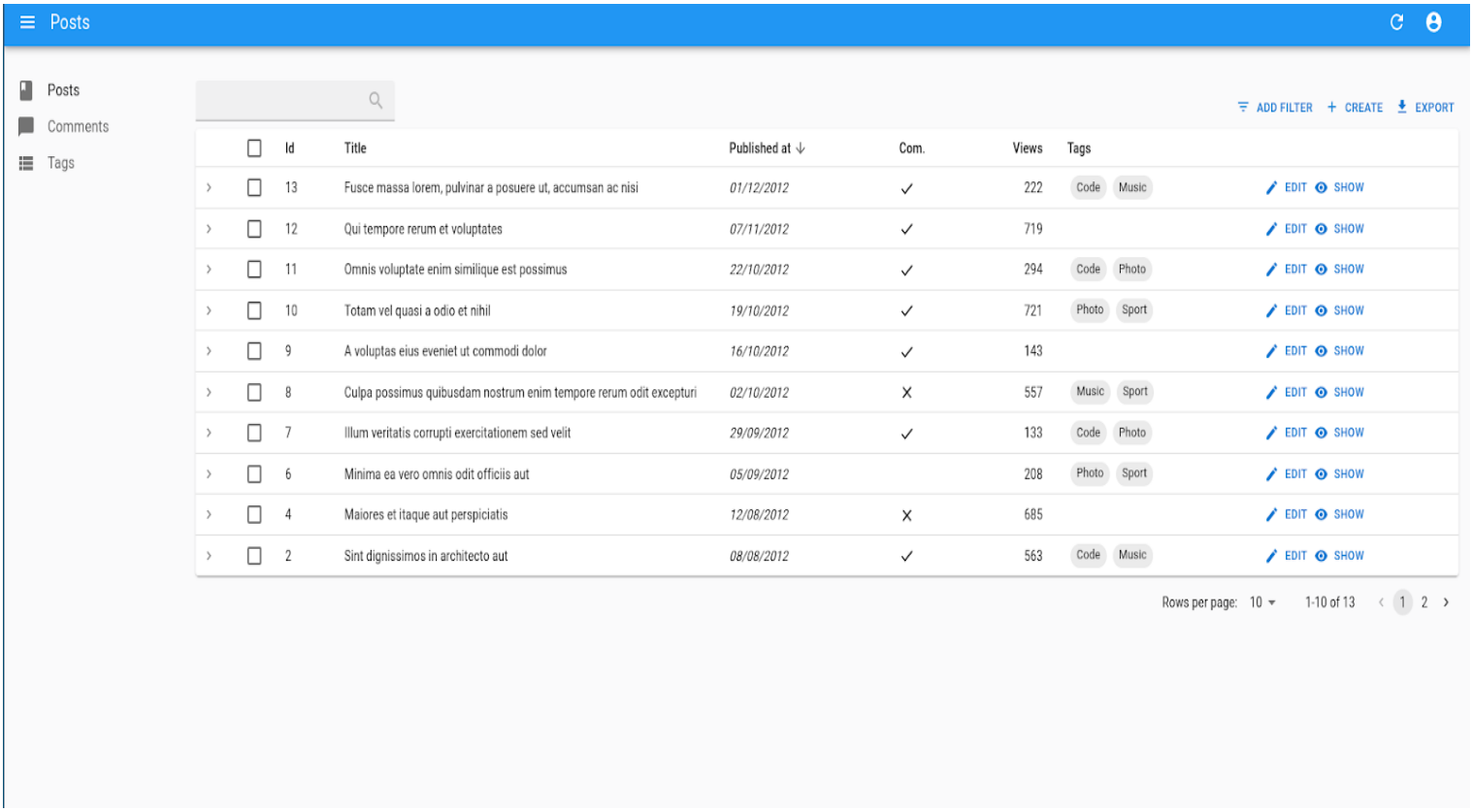
- AST Generation for Sunbird RC Config.
- Parser to auto generate code for X-Admin.
- Managing entities inside the registry using config.
- Implementing entities based dashboards.

Plan of Action

The objectives of the project are divided in the following parts :

- Creating a working Dashboard using react-admin.
- Generate the Abstract Syntax Tree (AST) of the source code in JSON format.
- On other hand, Implement the workflow to convert the JSON based AST again back to Code using JSON-ASTy
- Update the Registry from the Dashboard and it must make corresponding changes to both the JSON AST and The code.

First, We need to write the React-App with the help of react-admin and generate the dashboard as shown below with features and operations which we want to allow the admins to access from their side.



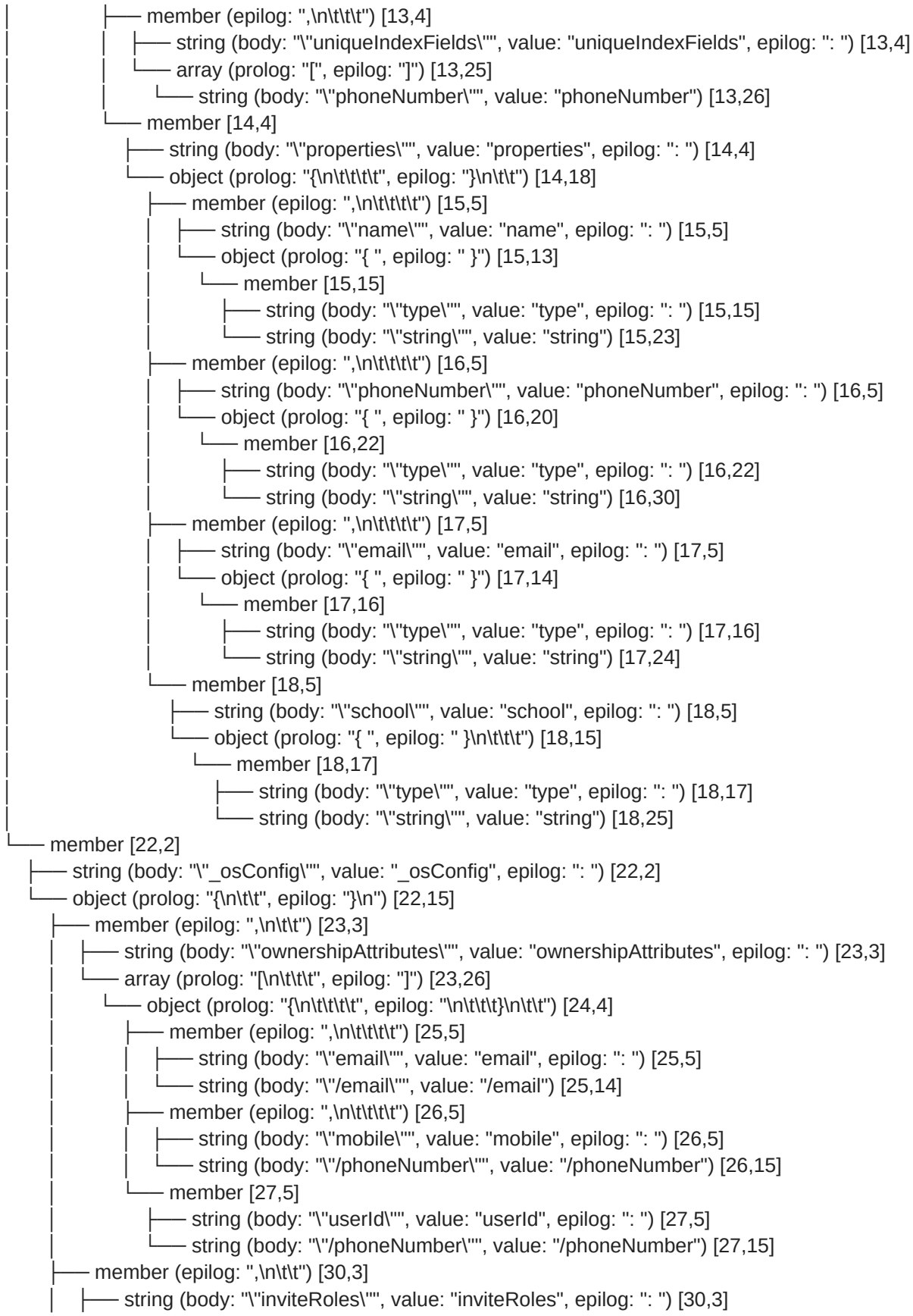
Once we are done with the app, We need to convert the source code into an Abstract Syntax tree (AST) which would be a JSON file.
We can use JSON ASTy for this purpose. It will allow us to convert JSON files to AST.
For Example : Let us generate the AST of the student config schema present in Sunbird RC.

```
let json = `{
    "$schema": "http://json-schema.org/draft-07/schema",
    "type": "object",
    "properties": { "Student": { "$ref": "#/definitions/Student" } },
    "required": ["Student"],
    "title": "Student",
    "definitions": {
        "Student": {
            "$id": "#/properties/Student",
            "type": "object",
            "title": "Studentschema",
            "required": ["name", "phoneNumber", "email", "school"],
            "uniqueIndexFields": ["phoneNumber"],
            "properties": {
                "name": { "type": "string" },
                "phoneNumber": { "type": "string" },
                "email": { "type": "string" },
                "school": { "type": "string" }
            }
        }
    },
    "_osConfig": {
        "ownershipAttributes": [
            {
                "email": "/email",
                "mobile": "/phoneNumber",
                "userId": "/phoneNumber"
            }
        ],
        "inviteRoles": ["anonymous"],
        "attestationAttributes": ["school"],
        "attestationPolicies": [
            {
                "property": "school",
                "paths": ["$.school"],
                "type": "MANUAL",
                "attestorEntity": "Teacher",
                "conditions": "(ATTESTOR#$.school#.contains(REQUESTER#$.school#))"
            }
        ]
    }
}`
console.log(`JSON (old):\n${json}`);
let ast = JsonAsty.parse(json)
console.log(`AST Dump (all):\n${JsonAsty.dump(ast)}`);
```

Output :

AST Dump (all):

```
object (prolog: "{\n\t", epilog: "}") [1,1]
├── member (epilog: ",\n\t") [2,2]
│   ├── string (body: "\"$schema\"", value: "$schema", epilog: ": ") [2,2]
│   └── string (body: "\"http://json-schema.org/draft-07/schema\"", value: "http://json-schema.org/draft-07/schema") [2,13]
├── member (epilog: ",\n\t") [3,2]
│   ├── string (body: "\"type\"", value: "type", epilog: ": ") [3,2]
│   └── string (body: "\"object\"", value: "object") [3,10]
├── member (epilog: ",\n\t") [4,2]
│   ├── string (body: "\"properties\"", value: "properties", epilog: ": ") [4,2]
│   └── object (prolog: "{ ", epilog: "}") [4,16]
│       └── member [4,18]
│           ├── string (body: "\"Student\"", value: "Student", epilog: ": ") [4,18]
│           └── object (prolog: "{ ", epilog: " } ") [4,29]
│               └── member [4,31]
│                   ├── string (body: "\"$ref\"", value: "$ref", epilog: ": ") [4,31]
│                   └── string (body: "\"#/definitions/Student\"", value: "#/definitions/Student") [4,39]
├── member (epilog: ",\n\t") [5,2]
│   ├── string (body: "\"required\"", value: "required", epilog: ": ") [5,2]
│   └── array (prolog: "[", epilog: "]") [5,14]
│       └── string (body: "\"Student\"", value: "Student") [5,15]
├── member (epilog: ",\n\t") [6,2]
│   ├── string (body: "\"title\"", value: "title", epilog: ": ") [6,2]
│   └── string (body: "\"Student\"", value: "Student") [6,11]
├── member (epilog: ",\n\t") [7,2]
│   ├── string (body: "\"definitions\"", value: "definitions", epilog: ": ") [7,2]
│   └── object (prolog: "{\n\t\t", epilog: "}") [7,17]
│       └── member [8,3]
│           ├── string (body: "\"Student\"", value: "Student", epilog: ": ") [8,3]
│           └── object (prolog: "{\n\t\t\t", epilog: "}\n\t") [8,14]
│               ├── member (epilog: ",\n\t\t\t") [9,4]
│               │   ├── string (body: "\"$id\"", value: "$id", epilog: ": ") [9,4]
│               │   └── string (body: "\"#/properties/Student\"", value: "#/properties/Student") [9,11]
│               ├── member (epilog: ",\n\t\t\t") [10,4]
│               │   ├── string (body: "\"type\"", value: "type", epilog: ": ") [10,4]
│               │   └── string (body: "\"object\"", value: "object") [10,12]
│               ├── member (epilog: ",\n\t\t\t") [11,4]
│               │   ├── string (body: "\"title\"", value: "title", epilog: ": ") [11,4]
│               │   └── string (body: "\"Studentschema\"", value: "Studentschema") [11,13]
│               ├── member (epilog: ",\n\t\t\t") [12,4]
│               │   ├── string (body: "\"required\"", value: "required", epilog: ": ") [12,4]
│               │   └── array (prolog: "[", epilog: "]") [12,16]
│               │       ├── string (body: "\"name\"", value: "name", epilog: ", ") [12,17]
│               │       ├── string (body: "\"phoneNumber\"", value: "phoneNumber", epilog: ", ") [12,25]
│               │       ├── string (body: "\"email\"", value: "email", epilog: ", ") [12,40]
│               │       └── string (body: "\"school\"", value: "school") [12,49]
```


member (epilog: ",\n\t\t\t") [13,4]
string (body: "\"uniqueIndexFields\"", value: "uniqueIndexFields", epilog: ": ") [13,4]
array (prolog: "[", epilog: "]") [13,25]
string (body: "\"phoneNumber\"", value: "phoneNumber") [13,26]
member [14,4]
string (body: "\"properties\"", value: "properties", epilog: ": ") [14,4]
object (prolog: "{\n\t\t\t\t", epilog: "}\n\t\t") [14,18]
member (epilog: ",\n\t\t\t\t") [15,5]
string (body: "\"name\"", value: "name", epilog: ": ") [15,5]
object (prolog: "{ ", epilog: " }") [15,13]
member [15,15]
string (body: "\"type\"", value: "type", epilog: ": ") [15,15]
string (body: "\"string\"", value: "string") [15,23]
member (epilog: ",\n\t\t\t\t") [16,5]
string (body: "\"phoneNumber\"", value: "phoneNumber", epilog: ": ") [16,5]
object (prolog: "{ ", epilog: " }") [16,20]
member [16,22]
string (body: "\"type\"", value: "type", epilog: ": ") [16,22]
string (body: "\"string\"", value: "string") [16,30]
member (epilog: ",\n\t\t\t\t") [17,5]
string (body: "\"email\"", value: "email", epilog: ": ") [17,5]
object (prolog: "{ ", epilog: " }") [17,14]
member [17,16]
string (body: "\"type\"", value: "type", epilog: ": ") [17,16]
string (body: "\"string\"", value: "string") [17,24]
member [18,5]
string (body: "\"school\"", value: "school", epilog: ": ") [18,5]
object (prolog: "{ ", epilog: " }\n\t\t\t") [18,15]
member [18,17]
string (body: "\"type\"", value: "type", epilog: ": ") [18,17]
string (body: "\"string\"", value: "string") [18,25]
member [22,2]
string (body: "\"_osConfig\"", value: "_osConfig", epilog: ": ") [22,2]
object (prolog: "{\n\t\t", epilog: "}\n") [22,15]
member (epilog: ",\n\t\t") [23,3]
string (body: "\"ownershipAttributes\"", value: "ownershipAttributes", epilog: ": ") [23,3]
array (prolog: "[\n\t\t\t", epilog: "]") [23,26]
object (prolog: "{\n\t\t\t\t", epilog: "\n\t\t\t}\n\t\t") [24,4]
member (epilog: ",\n\t\t\t\t") [25,5]
string (body: "\"email\"", value: "email", epilog: ": ") [25,5]
string (body: "\"/email\"", value: "/email") [25,14]
member (epilog: ",\n\t\t\t\t") [26,5]
string (body: "\"mobile\"", value: "mobile", epilog: ": ") [26,5]
string (body: "\"/phoneNumber\"", value: "/phoneNumber") [26,15]
member [27,5]
string (body: "\"userId\"", value: "userId", epilog: ": ") [27,5]
string (body: "\"/phoneNumber\"", value: "/phoneNumber") [27,15]
member (epilog: ",\n\t\t") [30,3]
string (body: "\"inviteRoles\"", value: "inviteRoles", epilog: ": ") [30,3]

```
│   └── array (prolog: "[", epilog: "]") [30,18]
│       └── string (body: "\"anonymous\"", value: "anonymous") [30,19]
├── member (epilog: ",\n\t\t") [31,3]
│   ├── string (body: "\"attestationAttributes\"", value: "attestationAttributes", epilog: ": ") [31,3]
│   └── array (prolog: "[", epilog: "]") [31,28]
│       └── string (body: "\"school\"", value: "school") [31,29]
└── member [32,3]
    ├── string (body: "\"attestationPolicies\"", value: "attestationPolicies", epilog: ": ") [32,3]
    └── array (prolog: "[\n\t\t\t", epilog: "]\n\t") [32,26]
        └── object (prolog: "{\n\t\t\t\t", epilog: "\n\t\t\t}\n\t\t") [33,4]
            ├── member (epilog: ",\n\t\t\t\t") [34,5]
            │   ├── string (body: "\"property\"", value: "property", epilog: ": ") [34,5]
            │   └── string (body: "\"school\"", value: "school") [34,17]
            ├── member (epilog: ",\n\t\t\t\t") [35,5]
            │   ├── string (body: "\"paths\"", value: "paths", epilog: ": ") [35,5]
            │   └── array (prolog: "[", epilog: "]") [35,14]
            │       └── string (body: "\"$.school\"", value: "$.school") [35,15]
            ├── member (epilog: ",\n\t\t\t\t") [36,5]
            │   ├── string (body: "\"type\"", value: "type", epilog: ": ") [36,5]
            │   └── string (body: "\"MANUAL\"", value: "MANUAL") [36,13]
            ├── member (epilog: ",\n\t\t\t\t") [37,5]
            │   ├── string (body: "\"attestorEntity\"", value: "attestorEntity", epilog: ": ") [37,5]
            │   └── string (body: "\"Teacher\"", value: "Teacher") [37,23]
            └── member [38,5]
                ├── string (body: "\"conditions\"", value: "conditions", epilog: ": ") [38,5]
                └── string (body: "\"(ATTESTOR#$.school#.contains(REQUESTER#$.school#))\"", value:
"(ATTESTOR#$.school#.contains(REQUESTER#$.school#))") [38,19]
```

So, Now we have the AST, we can make corresponding changes to the data from the dashboard itself, And it will update our JSON file.
If we want to go on a reverse method, We need to manipulate the generated AST and we can manually from the code itself, Create new nodes or delete them and this AST can be again unparsed into JSON file which would reflect on the dashboard. Either way, We can update the whole infrastructure of Registries.

Once we are set with this dashboard, We can think about implementing the authentication for it and writing the test cases.

Timeline

Week 01-02 Jun 20 - Jul 03

- Set Up JSON-Server, React-Admin, React-App, JSON-ASTy.
- Testing Config Schemas (Probably on Postman)
- Creating UI for the Dashboard and Implementing the decided data fields based on the config schemes.

Week 03-04 Jul 04 - Jul 17

- Getting the AST Trees using the appropriate tool like JSON-ASTy.
- Updating the Config Schemas from the Code itself to see if it works.
- Updating the Data Registry from the Dashboard and reflecting the changes into the JSON config.

Week 05-06 Jul 18 - Jul 31

- Applying Authentication to the Dashboard.
- Writing appropriate test cases.
- Adjusting unwanted bugs in case they occur.
- Wrapping up the project.

Relevant skills and motivation

What are your languages of choice and how do they relate to the project? What is your motivation to apply for this project?

I have been looking at open source software for almost a year and have personally used many of them for very basic to advanced tasks. With time, I started contributing to those projects which were aligning with my technical abilities. I have worked on many Web Development projects and did many small documentation fixes. I have worked on React and Typescript based projects. I also have worked for LLVM Compiler Toolchain in the Flang Compiler of Fortran where I have applied semantic checks. So, I can technically relate close enough with the project. I do not know

everything so there is a very good scope of learning about the technologies involved and gaining experience with the ones which I already know.

Personal Information

I am **Harshil Jani**, Student at National Institute of Technology (**NIT Surat**) pursuing my Bachelors of Technology majors in **Electronics and Communication Engineering** along with my minors in **Computer Science Engineering**. I was introduced to the world of programming and software development in my first year. Since then, I have been very enthusiastic about deep diving into various fields of Computer Science. The areas that capture my interests are Data Structures, Algorithms, Operating Systems. For most of my programming journey, I have worked primarily with **Web**-based technologies, **C/C++** programs and I have recently been diving into **Compiler** Design and into **Rust** Programming Language.

I am an Executive Member at **ACM** (Association for Computing Machinery) NIT Surat Students Chapter. Outside my university, I am part of open source communities and my role in them is to help people with Linux related issues. I mostly help people with ricing their Operating Systems in those communities. I have been part of multiple open source events conducted related to documentation (GWOC) or contributing to some of the projects (WOC 2.0, GSSOC, Hacktoberfest, SWOC). I am also selected as a Google Summer Of Code (GSoC) contributor for this year with CERN-HSF and working on Data Visualization with D3.js .

Communication

I’m flexible with my schedule and have inculcated the habit of working at night. I can work full-time on weekdays and am usually available between

3PM(IST) to 2AM(IST) on weekends. From 25th of July 2022, My next Academic Year will be continued at university so I will have to migrate to my university a week before. So, I have adjusted the working timeline accordingly. Other than that, I am committed to GSoC program and my project is medium sized of around 175 hours spread for 12 weeks from 13th July. So, I will also have to spare time for it.
I will make sure to inform the mentors beforehand regarding any time changes or emergency in case they arise with suitable details.

Post Event

If there are things left unimplemented, I will try to complete them even after completion of the coding time period. If there are any future objectives that are required for this project or any major improvements are expected which falls under my technical ability to work on, I would be more glad to work for it even after the event. After All that is what open source means.

Some Open Source Contributions

- **UCI Web Channel UI Changes**
  Issue : https://github.com/samagra-comms/uci-web-channel/issues/10
  PR : https://github.com/samagra-comms/uci-web-channel/pull/11
- **DSC JSS NOIDA Resource Website UI Changes**
  Issue : https://github.com/DSC-JSS-NOIDA/QuickLearn/issues/110
  PR : https://github.com/DSC-JSS-NOIDA/QuickLearn/pull/124
- **Gitpod UI Fixes (TypeScript)**
  Issue : https://github.com/gitpod-io/gitpod/issues/7991
  PR : https://github.com/gitpod-io/gitpod/pull/8875
- **LLVM Flang (FORTRAN Compiler FrontEnd)**
  Commit : https://github.com/llvm/llvm-project/commit/bea53eead1de84a28affc6a7cbf88f87a258fed4
  Diff Patch : https://reviews.llvm.org/D119141

And many more such contributions…