Asish Kumar
xyz@gmail.com
City, state, country
Timezone: IST (UTC+5:30)

Conversion Rate

GSoC Project proposal for CHAOSS

Personal Details and Contact Information

- Github: officialasishkumar
- Email:
- University: Bharati Vidyapeeth's College of Engineering, New Delhi
- Time-Zone: IST (UTC + 5:30)
- Address:
- IRC nick: Asish Kumar
- Linkedin: https://www.linkedin.com/in/asish-kr/
- Portfolio:

Synopsis

This project's main aim is to build a mechanism to analyse how a user who interacted with a project in any form, may it be writing a simple comment on an issue or pull request or just attending an event is becoming a contributor. It also aims to understand how these contributors impact the overall growth of the community (or project). The definition of contributor here is anyone who does some work for the community, may it be writing code or reporting issues.

I would be implementing this feature in augur repository and will be working under mentor Sean Goggins.

Benefits to the Community

In the broader picture of FOSS in general, this project will allow small organisations who are working on a new Open source software gain contributors and understand where they can find and how they can find new contributors by looking at the stats and visualization data. This is important because in the longer run every codebase becomes large and the founders cannot maintain or dedicate enough time for the future development of the project. This will cause a

technical backlog and the code might not be sync with the current standards. Helping these orgs find new contributors through CHAOSS will help them to make their project survive and grow in a steady way.

This will also allow CHAOSS to grow their community and gain new members, whether it's the people who will use the application or new contributors.

Current Status of the Project

The current status of the project is incomplete and I will need to implement everything from scratch. However, I will be using some of the existing tables and extract information from them based on the needs of this feature. One example of that is using the `Contributors` and `Contributors_aliases` to fetch the active contributors in the repository. These tables will allow me to access the users currently contributing in some form to the repository and give them a label to understand their hierarchy in the contributor metrics.

Goals

“Conversion Metrics” Project which will help the project maintainers to understand how often a user who interacts with the community in any form is becoming a seasoned contributor. They should get an easy to understand metric such as a percentage on how many users who attended an event became a contributor and how many of them stuck around for lets say X number of days or created Y number of issues and pull requests.

The stretch goals would include advancing the work of CHAOSS metric models working group. This involves adding definition and implementation around the new contributors metric. This would also include implementing the existing models whose definitions are provided already.

In the later part of the project, I would also be assisting in implementing a website for viewing these metrics.

Deliverables

Deliverable 1

Implementing an initial setup which would start from D0 (users who joined any event hosted by the community and have filled a survey form), further moving to D1 (users who are watching or forked any repository in the organization), then to D2 (which would

involve users who have created an issue, pull request or wrote any comments) and finally to D3 (which will have all the users whose pull request got merged).

Deliverable 2

Implementing specific logic for moving someone from Dx to Dy, where x and y are numbers from 0 to 3.

This part of the coding phase would also include completing the conversion rate metrics and starting to work on extended goals which includes advancing the work of the CHAOSS metrics models working group and assisting in development of a website for showing these metrics.

Deliverable 3

This part of the coding phase would include writing unit tests and making sure everything works after merging all my pull requests. This phase would also include writing documentation and any blogs that would include how this feature was implemented and what are the challenges faced.

Expected Results

- Conversion metric that will tell the project maintainers on how often a user becomes a seasoned contributor.
- Improved CHAOSS metrics models working group with more implementation using a lot of metrics.
- Website for displaying the CHAOSS metrics for selected repositories.

Approach

The first part is to understand the different levels we will assign the users who interacted with the organization. Here are all the definitions for these levels

- D0 will contain all the users who attended a live event.
- D1 will contain all the users of D0 who went to the repository and either they clicked watch or forked the repository.
- D2 will contain all the users of D1, who created an issue, pull (or merge) request, or just commented or did a review.
- D3 will contain all the users from D1 whose pull request got merged or have a high probability of getting merged.

Now, lets understand how i will be implementing these levels:

For D0, we can have an online survey of some form where we can ask the attendees of the event to fill out necessary details like github/gitlab username, email and Full Name along with their country they are residing in and age. These data will directly be inserted into a table whose schema is as follows:

These values must be provided in the form that the attendees have to fill. These are important because it will help us to map the users in github and gitlab when they do some activity in the repository. Age and country will allow the organization maintainers to understand which age and country group are more actively becoming a contributor so that they can focus their campaign in that direction. While event time is needed to understand how many days it took for the attendees to check out the codebase of the organization.

I will provide some endpoints to easily fill this table without worrying about the table schema or how things are implemented under the hood.

Here is an example implementation, that will fetch the data submitted from Google Form and insert it into database without any manual work

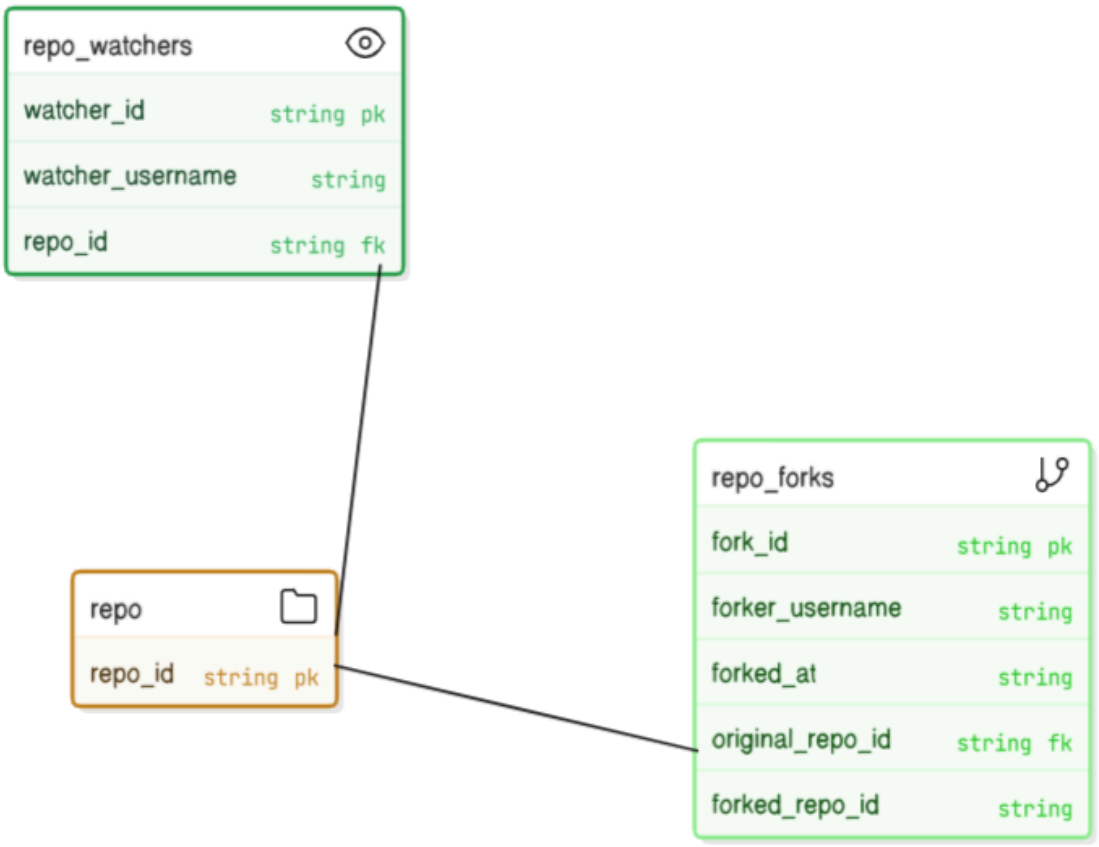
```
def get_sheet_data():
    scope = ["https://spreadsheets.google.com/feeds",
"https://www.googleapis.com/auth/drive"]
    creds = ServiceAccountCredentials.from_json_keyfile_name("credentials.json",
scope)
    client = gspread.authorize(creds)
    sheet = client.open_by_key(SPREADSHEET_ID).worksheet(SHEET_NAME)
    return sheet.get_all_records()

def insert_into_db(record):
    conn = psycopg2.connect(PG_CONN_STR)
    cur = conn.cursor()
    query = """
    INSERT INTO event_attendees
      (full_name, email, company_name, github_username, gitlab_username, age,
country, event_time)
    VALUES (%s, %s, %s, %s, %s, %s, %s, %s)
    """
    data = (
        record.get("full_name"),
        record.get("email"),
        record.get("company_name"),
        record.get("github_username"),
        record.get("gitlab_username"),
        record.get("age"),
        record.get("country"),
        record.get("event_time")
    )
    cur.execute(query, data)
    conn.commit()
    cur.close()
    conn.close()

@app.route('/sync', methods=['GET'])
def sync():
    try:
        records = get_sheet_data()
        for record in records:
            insert_into_db(record)
        return jsonify({"status": "success", "records_synced": len(records)})
    except Exception as e:
        return jsonify({"status": "error", "error": str(e)}), 500
```

After this step, our D0 will be over and now we can move to defining and connecting D1 with D0.

We can leverage, github and gitlab username that we got from D0 to find out how many attendees of a particular event went on to github or gitlab to interact with the codebase.

I will be using forks and list watchers endpoint provided by Github API to get all the people who have forked or are watching the repository from `event_time` (which we can get from from `event_attendees` table).

I will be creating two new tables that will store the users who have forked or are watching the repository.



The `forked_at` field will help us to determine whether the user started interacting with the repository after the event or before the event. Since the github API does not provide the time the user opted for watching the repository, we cannot have that there. We will use both these tables to find out which users who are in D0 are moved to D1 and when exactly did they interact with the repository. These tables will allow us to define the conversion rate from D0 to D1 along with other information like what's their age and country to better analyze the data.

Here is an example implementation of how i will fetch these information using the github API

```
def list_forks(repo_id, token=None):
    url = f"https://api.github.com/repositories/{repo_id}/forks"
    headers = {}
    if token:
        headers["Authorization"] = f"Bearer {token}"
    response = requests.get(url, headers=headers)
    if response.status_code == 200:
        forks = response.json()
        return forks
    else:
        print(f"Failed to retrieve forks: {response.status_code}")
        return None


def list_watchers(repo_id, token=None):
    url = f"https://api.github.com/repositories/{repo_id}/subscribers"
    headers = {}
    if token:
        headers["Authorization"] = f"Bearer {token}"
    response = requests.get(url, headers=headers)
    if response.status_code == 200:
        watchers = response.json()
        return watchers
    else:
        print(f"Failed to retrieve watchers: {response.status_code}")
        return None
```

Now that we have all the users who are in D1 we can now move to D2. Let's understand the implementation of this level.

We need to map all the issues, pull requests and comments made by the users who are in D1.

I will use the issue table to see whether any of the users in D1 have created an issue and if they created it after the community event has happened. Also, a good add on to this would be to give a priority score of the issue to understand how impactful that issue is which can be calculated with this formula

```
Priority Score = (2 × 👍 reactions) + (# comments) + (3 × # participants) + (5 × if labeled 'critical')
```

For pull requests, we can do the same, that is find the username and time the pull request is created from the existing table and then figure out a formula for priority score.

This equation is just an example and I will be changing based on the mentors feedback during the GSoC program.

This priority score will help us to find the impact a user creates when they are in D1 level. This priority score would be stored in the `contributors` table.

For finding comments, we can leverage the existing tables to figure out and bridge a connection on whether a particular user (who is part of D1) commented on any pull request or issue using a sql query.

Moving to D3, this will be the final level which will have users from D2 whose pull requests are merged which we can get from the column `pr_merged_at` from the table `pull_requests`. The users whose `merge_probability` are more than a certain threshold will also be considered in D3 level.

I will also add a metric to get the individual impact of a user in a community by the number of pull requests (open and merged), number of issues created and the number of PR reviews done. This will allow me to create a leaderboard in the metric website.

All the metrics for these levels and the underlying logic for pushing someone to a new level will be defined in the `augur/api/metrics/contributors.py` file. I would also be adding new files based on the mentors feedback.

The code for fetching the people who are watching or have forked the repository will go in `augur/api/tasks/github` and `augur/api/tasks/gitlab` folders.

There will be an API endpoint for fetching the event attendees which will go in `augur/api/routes` folder.

Improving the CHAOSS Metrics Model Working Group

I will be adding the following metrics to better understand the new contributor metric that will be implemented

- Which demographic location has the highest conversion rate?
- Which age group has the highest conversion rate?
- How impactful a new contributor is based on the priority_score we have defined?
- How many days does it take for a user to move from Dx to Dy to better understand how effective our documentation is?
- The impact in the community of a particular contributor who has reached Dx level decided to leave?

Apart from adding these new models, I will also be implementing the existing ones whose only definitions are provided after discussing it with the mentors.

Metric website

The metric website will serve the purpose of understanding the contributor metrics result in a better way. Here are some of the things that the website will serve:

- Number of users in each level.
    - A pie chart to show the age and demographic location of these users.
    - For this I will be using the code snippet from https://adminkit.io/docs/charts/pie-chart
- Conversion rate of people from Dx to Dy
    - Filter to choose the country and age of the users and get results based on that
    - A bar chart to show the average time to go to a particular level.
- A leaderboard showing the maximum impact of a user (who is in D3 level) to the community in a positive way
    - This impact will be calculated based on the number of pull requests (both merged and open), number of issues reported, and the number of review comments. I will be coming up with a formula to find the best number that will decide this leaderboard.
- A board displaying the impact of the event based on the number of contributors it retained. Other factors that will help in the calculation of this board would be the impact of a contributor in D3.

There will be a filter to choose the organization that we want the contributors metric for. Including this, every board (or table) that I discussed above will have a sort feature.

Timeline

Period	Task
After proposal submission [April 10 - May 6]	- Understand every metric and run them locally - Understand how every table is connected and figure out the ones which I will need for this project. - Draft a detailed plan for how I will be implementing the code for phase 1 of the project - Explore other projects under CHAOSS
Week 1 and 2 [May 6 - May 27]	- Write a final plan on the phase 1 of the project with all the implementation details. - Read all the necessary documentation - Decide the new tables schema
Week 3 and 4 [May 27 - June 8]	- Create new tables for storing the event attendees information - Implement D0 by providing different API to find the users who showed interest. - Implement all the metrics required in D0
Week 5 and 6 [June 9 - June 22]	- Write metrics and API endpoints for D1, finding the users from D0 who are watching or have forked the repository. - Document all the implementation details and which external apis are used. - Start with the implementation for D2

Week 7 and 8 [June 23 - July 6]	- Completing the implementation for D2 and all the conversion factors that will move a user from D1 to D2. - Move on to implementing D3 which will involving deciding on a threshold of merge_probability
Week 9 and 10 [July 7 - July 20]	- Fully complete the implementation for D3 and all the levels. - Write Unit tests and documentation for all the changes made. - Fix any bugs or bottlenecks. - Write 60% of the models for contributor metrics.
Week 11 and 12 [July 21 - August 3]	- Complete the development of models of contributor metrics. - Implement any other models whose definitions are already provided or any new ones once approved by the maintainers. - Start working on deployment of this metric in a publicly viewable website.
Week 13 and 14 [August 4 - August 17]	- Complete the deployment of the website for viewing the metric. - Write test cases as required. - Any documentation that is left over.
Week 15 and 16 [August 18 - September 1]	- Fix any bugs. - Blog of my whole journey of the GSoC program which will be published on my github pages. - Write final work that will be submitted for review 2.

About Me

I'm Asish. I have done GSoC 2024 with TARDIS-SN and LFX 2024 (term 2) with KCL language. In GSoC 2024 I have worked with Python. I have a lot of experience in contributing to open source projects and I love tackling tougher problems. My expertise is in Python, Golang, Javascript and C++. I think my strong open source experience would be a great asset in the success of this project.

Some of my past contributions made to CHAOSS are:

- Issues
    - https://github.com/CHAOSS/augur/issues/3051
    - https://github.com/CHAOSS/augur/issues/3050
    - https://github.com/CHAOSS/augur/issues/3081

- Pull request
    - https://github.com/CHAOSS/augur/pull/3062
    - https://github.com/CHAOSS/augur/pull/3064
    - https://github.com/CHAOSS/augur/pull/3084

I discovered CHAOSS while randomly browsing through the GSoC page. It caught my attention because it focuses on helping open source organizations grow through visualizations and metrics which I thought was cool stuff. I'm planning to make meaningful contributions to CHAOSS by enhancing the current metrics and providing users with more features to better understand how their organizations are growing.

Throughout the summer, I won't have any major commitments, so I'll be able to dedicate 7–8 hours a day to this project, totaling around 40 hours each week. I plan to log my work using Linear—a tool that helps track the problems I'm working on, their broader context, and when they're completed, along with all relevant details. Additionally, I'll be documenting my code and sharing my journey on a personal blog every two weeks, maybe hosted on github pages.

I'd love to stay active in the CHAOSS community even after GSoC. While I may not be able to consistently contribute new features post-GSoC, I'd be happy to help with code reviews, merge requests, or anything else that supports CHAOSS in reaching more users and communities.