Google Summer of Code 2021

<u>Personal Information</u>

Identification -

➜ **Name -** Saurav Shrivastav

➜ **Nationality -** Indian

➜ **Location -** Pune, Maharashtra, India

➜ **Timezone -** IST (UTC +5:30)

Contact -

➜ **Email** - sauravsrivastav103@gmail.com

➜ **Phone** - +91 8830588434

➜ **Github** - Saurav-Shrivastav

➜ **Slack** - Saurav Shrivastav

➜ **Alternate Email -** sshrivastav_be19@thapar.edu

Educational Information -

➜ **Institute -** Thapar Institute of Engineering and Technology

➜ **Degree -** Bachelor of Engineering

➜ **Major -** Computer Engineering

➜ **Current year -** Sophomore (2nd Year)

About Me

Educational -

I am a second-year undergraduate student pursuing my Bachelor's degree in Computer Engineering at TIET, Punjab, India. I was introduced to the ways of programming in my high school when I started learning Java. Since then, I've been trying out various technologies by taking up new projects, participating in hackathons, and contributing to Open Source.

My complete Academic Resume can be viewed here.

Volunteer Experience -

- [Google Developer Student Club](#) - Core Member
  - It is a group of passionate developers and designers. This is where I was introduced to the world of Open Source.
  - At DSC, I got a chance to develop software for thousands of students and configure our servers and deploy our apps into production.
  - Learned how to solve issues efficiently and also how to collaborate better with my team.
- [Microsoft Learn Student Chapter](#) - Core Member
  - Learned a lot of time management skills and the ability to deliver, punctually, under pressure.

Freelance Experience -

- CYO gyms
  - Implemented an API with Django-graphene.
  - Created a subscription-based Django channels application.
- AnalyticWare
  - Implemented GAuth and integrated a payment gateway for the project.

Open Source Experience -

- [BotsFramework](#) -
  - Implemented continuous delivery using TravisCI to build the plugin upon a new release.

- ○ Scraped websites like Medium.com, Dev.to, etc., and built a minimal flask server that provides a JSON response of a list of articles for the particular tag/keyword to be searched.
  - ○ Integrated the scraper flask service to build slack and discord bots that send daily articles onto the channels.
  - ○ Worked as the project maintainer and managed various issues and pull requests for the project.
- ● [DSC Official Website](#)
  - ○ Migrated the project to pipenv.
  - ○ Worked on some feature endpoints and models.
  - ○ Documentation updates

Motivation -

I have worked on various projects where we used PostgreSQL as a production database. Contributing to an organization whose tools are used by people around the globe is the biggest motivation of all.

I have been working with Django and Python for a considerable amount of time. It acted as one of the reasons that motivated me to select PostgreSQL as my GSoC organization. Contributing code to a big project like PostgreSQL will be a great learning opportunity and advancement to my career.

GSoC Project

Improve pgeu-system for Conference Management

Pg-eu system is an infrastructure developed and maintained by Postgres for managing non-profit organizations along with an extensive conference management system. It contains a set of features such as invoices, administration of conferences, memberships, and elections, along with social media integration. Some improvements could be made to the system, which is being proposed, such as adding HTML emails and adding a generalized form-building capability. The GitHub issues for the same are #52 and #53.

The project is divided into 2 major phases -

Phase 1 -

> Support HTML emails
> Open and link tracking of emails

Phase 2 -

> Generalized form/survey system

Measurable Outcomes -

- ❖ Support for HTML emails
- ❖ Setup email tracking
    - ➢ Add tracking information to emails
    - ➢ Handle email opens and link clicks
    - ➢ Make tracking data available to administrators
- ❖ Build a form/survey system
    - ➢ Enable administrators to craft surveys to send to members and others
    - ➢ Generate Pdf and CSV reports for forms
- ❖ Add documentation for the newly incorporated features.
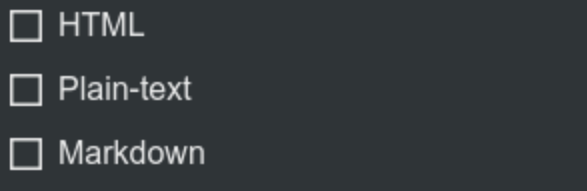
Project Details -

Phase 1 -

- **Support for HTML emails**
  - The current system for email supports sending text-based emails only. A section of code present in mailqueue/util.py uses the Django template system as a preparation which will make it easier to add HTML emails

```python
def send_template_mail(sender, receiver, subject, templatename, templateattr={},
attachments=None, bcc=None, sendername=None, receivername=None,
suppress_auto_replies=True, is_auto_reply=False):
    send_simple_mail(sender, receiver, subject,
                     template_to_string(templatename, templateattr),
                     attachments, bcc, sendername, receivername,
                     suppress_auto_replies, is_auto_reply)
```

  As of now, it renders text to text only and just uses the templating to substitute variables in the text.
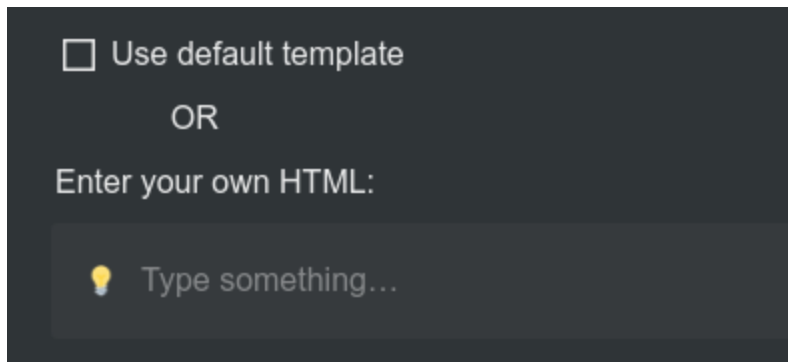
  - The mail sending flow already uses MIME, which is most commonly used for sending HTML and plain-text emails. Since all email clients do not display HTML content by default, and some people choose only to receive plain-text emails, a plain-text alternative will also be included for HTML messages.
  - The functions for sending emails will be modified and support will be added to send HTML and Markdown based emails.
  - After implementing HTML emails, the WEB UI for sending emails will be modified and it will enable the user to switch between sending HTML, plain-text, or markdown based mails.

  For HTML emails, the user can select between using a default template or

adding their own HTML.

- **Setup email tracking**
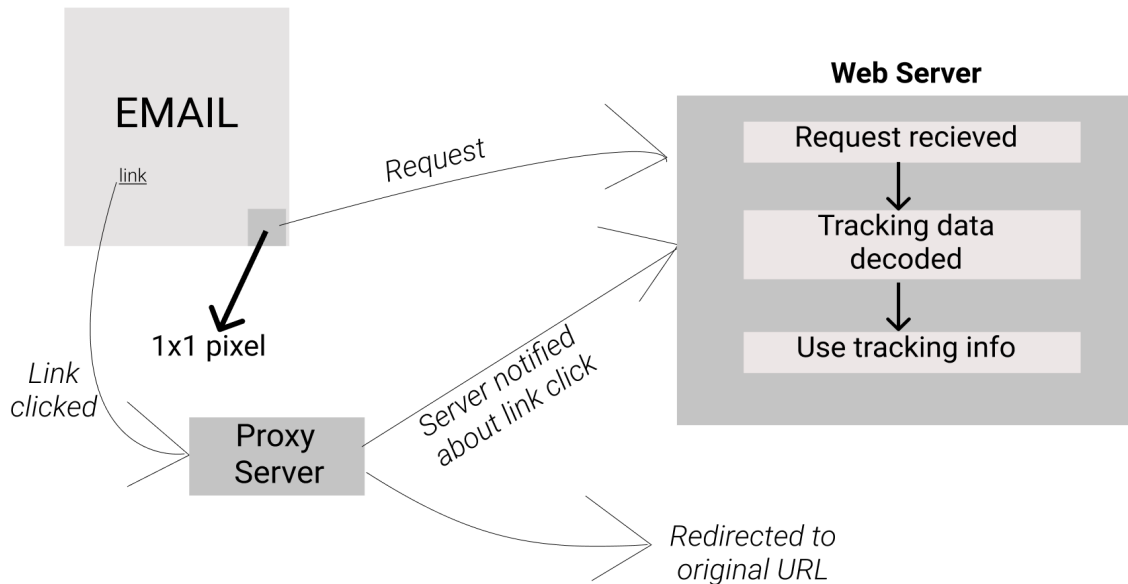 - Adding tracking information to emails:
    The approach will be to add a small 1x1 transparent pixel at the end of each email. When a user opens an email, the email client will load the pixel by making a GET request. The web server serving the request will then record the open and notify the sender of the email.
    To track links, the link will be rewritten to change their destination to a proxy. Once a user clicks on the link, the proxy redirects the user to the real link and notifies the sender of the email.
  - Handling email opens and link clicks:
    Once a user opens an email or clicks on a link, the email client will send a request to the encoded URL. The web server will receive such a request, where the tracking information will be decoded.
  - The tracking data can then be used for various purposes, generate reports in pdf, csv files or display the information on the panel.
 - Diagram of the approach -

EMAIL

link

Request

1x1 pixel

Link clicked

Proxy Server

Server notified about link click

Redirected to original URL

Web Server

Request recieved

↓

Tracking data decoded

↓

Use tracking info

Can also be viewed on figma.

Phase 2 -

- Build a **form/survey system -**
  - The idea is to have a generalized form-building capability which administrators could use to craft surveys to send to members and others.
  - Main objective of this phase is to build a reusable django app that can be used across Postgres projects for building and managing surveys and forms.
    The survey system has been divided into 3 major sections as depicted in the following diagram:

Postgres Survey System

The form builder   The forms   Responses and Reports
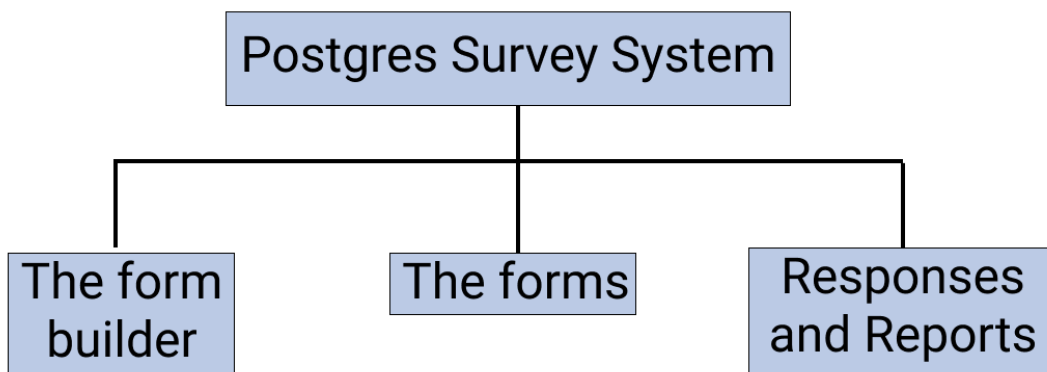
- ○ *Form builder -*
  It will provide an interface to build forms. The administrator will be able to perform the following operations -
    - ■ Give a title and description to the form.
    - ■ Activate/deactivate the form.
    - ■ Add questions.
    - ■ Send customized emails to form responders.
    - ■ Choose between different answer fields for each question - TextField, TextBox, Radio button (single choice), Check box, Date, Time.
    - ■ Mark a question as required/not required.
    - ■ Give questions categories.
- ○ *The form -*
  Once the form is saved, a link will be generated, which can then be sent to members or added to web pages.
- ○ Responses and Reports -
    - ■ All the responses will be visible to the administrator at all times.
    - ■ Generation of CSV, PDF reports will be made possible.

Proposed Timeline -

May 17 - June 7 Community Bonding Period	● Interact with the mentors and set up feedback loops. ● Continue to refine the plans for the project in consultation with the mentors, I have mentioned the path and way for the solutions in the proposal but they are flexible and can change as per the common view with the mentor. ● Get involved with other members of the community. ● Extensive research on implementations and methodologies for the set goals to make the solutions easier

Approximate division -

- Phase 1 - 3 weeks
- Phase 2 - 6 weeks
- Buffer time - 1 week

PHASE - 1

Week 1 & Week 2 - June 7 - June 19	• Add support for HTML emails • Begin setup for email tracker
Week 3 > June 21 - June 26	Complete email tracker

PHASE - 2

Week 4 > June 28 - July 3 Week 5 > July 5 - July 10	Implement the form builder.
Week 6 > July 12 - July 17 Week 7 > July 19 - July 24	Work on form generation.
Week 8 > July 26 - July 31 Week 9 > Aug 2 - Aug 7	Record Responses and Generate Reports features.
Week 10 > Aug 9 - Aug 16	Buffer Period

NOTE: The documentation updates (if any) will go on parallely with feature work.

Apart from this, I will be maintaining a 2-part blog (for each phase), writing about my work and the entire workflow
I have also kept some buffer days in my time, giving me time to complete any delayed tasks.

Availability -

During the GSoC period, I can spend around 40-45 hours per week on the project. I don't have any pre-planned vacations or plans during the break and will be available full time.

My college will reopen somewhere around mid-August, and I can devote around 30-35 hours per week once that happens. I intend to complete most of the work before my college reopens and will use the buffer days in case of any backlogs.

I shall keep my status posted to all the community members on a weekly basis and maintain transparency in the project.

After GSoC

For me, the open-source contribution is not just restricted to Google Summer of Code. I would love to keep working on the developments and enhancements. Here are some of the ideas that I would love to implement after the entire GSoC timeline -

- Automate Python-workflow with Pre-commit hooks, use black and flake8 for formatting and checking code.
- Add features to the survey system -
  - Produce better reports with charts and graphs
  - Add more customisation