Python Software Foundation GSoC Team

Resolve bugs and usage issues

About Me

- Name: Diwash Dahal (**diwash007**)
- University: Tribhuvan University
- Program: BSc CSIT (4 year/ 8 semester)
- Expected graduation year: 2023
- Timezone: Nepal (GMT+5:45)
- Website: https://diwashdahal.com.np
- Github: https://github.com/diwash007

I am a sixth-semester Bsc CSIT student from Jhapa, Nepal. I have been interested in technology since my school.
I have acquired good programming knowledge by learning technologies like HTML, CSS, JS, PHP, Python, Flutter, etc. I have built a few projects with the technologies I learned. They are listed on my website and on Github.
I have been actively contributing to the Python blogs project of the Python GSoC Team.

Code Contribution to Python GSoC blog

S.N.	Pull Request	Solved Issue	Description
1	#440	#432	Add user profile information change feature
2	#442	#363	Add suborg admin to suborg application
3	#446	#429	Add feature to see added mentors by suborg-admin
4	#449	#380	Randomize students blogs
5	#450	#367	Show suborg of users on user list
6	#452	#366	Add feature to export mentor emails
7	#454	#453	Add the suborg-admin role to admins on application acceptance

Project Information

- Suborg:

  Python Software Foundation GSoC Team

- Mentor:

  Botanic (Matthew Lagoe)

- Skills Required:

  Django, HTML, JS, and Python

- Difficulty:

  Intermediate

- Project Abstract:

  This project aims to add enhancements, solve issues and improve the user interface of the **Python Software Foundation GSoC Blog** platform. Currently, there are several issues listed on the issue tracker of the PSF GSoC Blog. This project aims to solve those issues and improve the usability of the blog platform for admins and students.

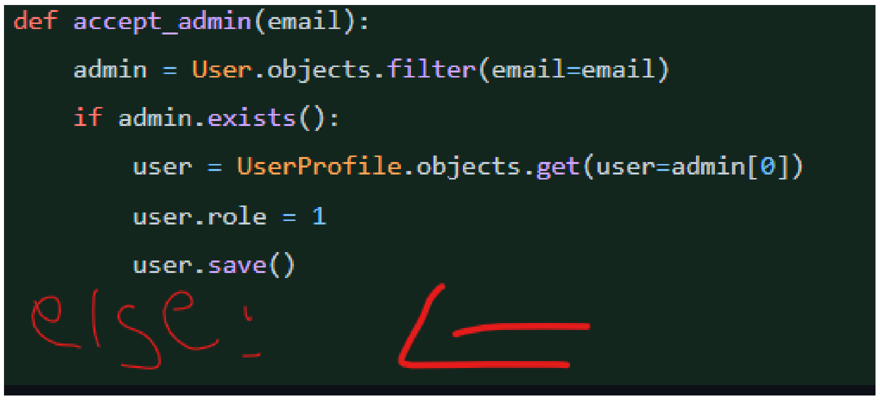
Project Details

I am choosing **medium** size project. I.e **175hrs**. I think that will be enough to fix the listed issues. We can extend the project to 350 hours if required.

Since the project is about resolving bugs and usage issues, I will be working on solving different issues on the platform. I will proceed as follows with priority top to bottom:

1. **suborg admin user profile #453 bug Highest Priority - Active**

   Currently Suborg admins aren't given the suborg-admin User profile. I will fix that bug so that the admins get the required role. I am currently working on this issue so the only thing that is left to do on this issue is to handle if the admins haven’t been registered onto the system.



```
def accept_admin(email):
    admin = User.objects.filter(email=email)
    if admin.exists():
        user = UserProfile.objects.get(user=admin[0])
        user.role = 1
        user.save()
else:
```

2. Users registering directly onto the system creates a blank profile #457 bug Priority Issue

User registering into the system without associating with any Suborg creates a blank UserProfile. This can be eliminated by checking if the user is associated with any Suborg or not.

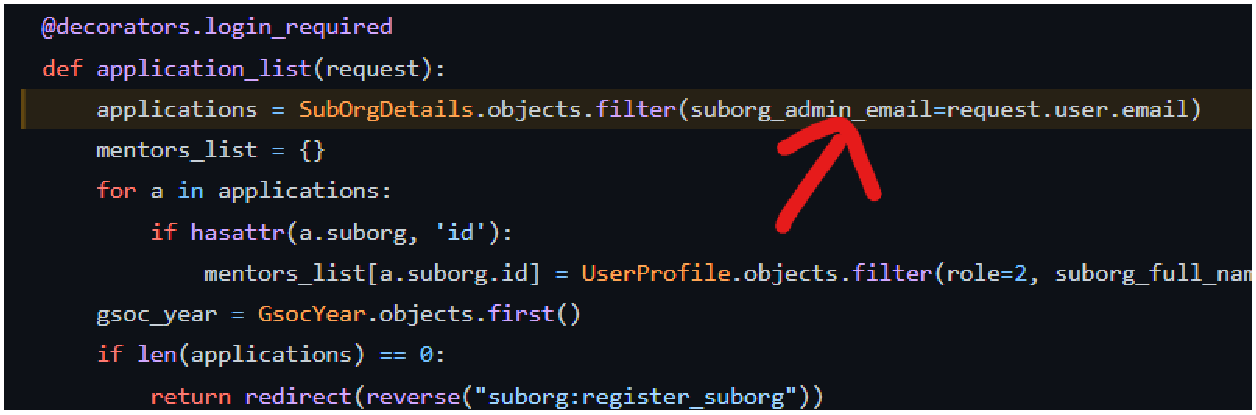
Currently, the profile is saved as follows:

```
    profile = UserProfile.objects.create(
        user=user,
        role=self.user_role,
        gsoc_year=self.gsoc_year,
        suborg_full_name=self.user_suborg,
        reminder_disabled=reminder_disabled,
        github_handle=github_handle,
    )
    if self.user_role != role.get("Student", 3):
        profile.save()
        return user
```

I will add a condition to check if the user is associated with any Suborg or not.

3. **changing email breaks suborg applications #433 bug**

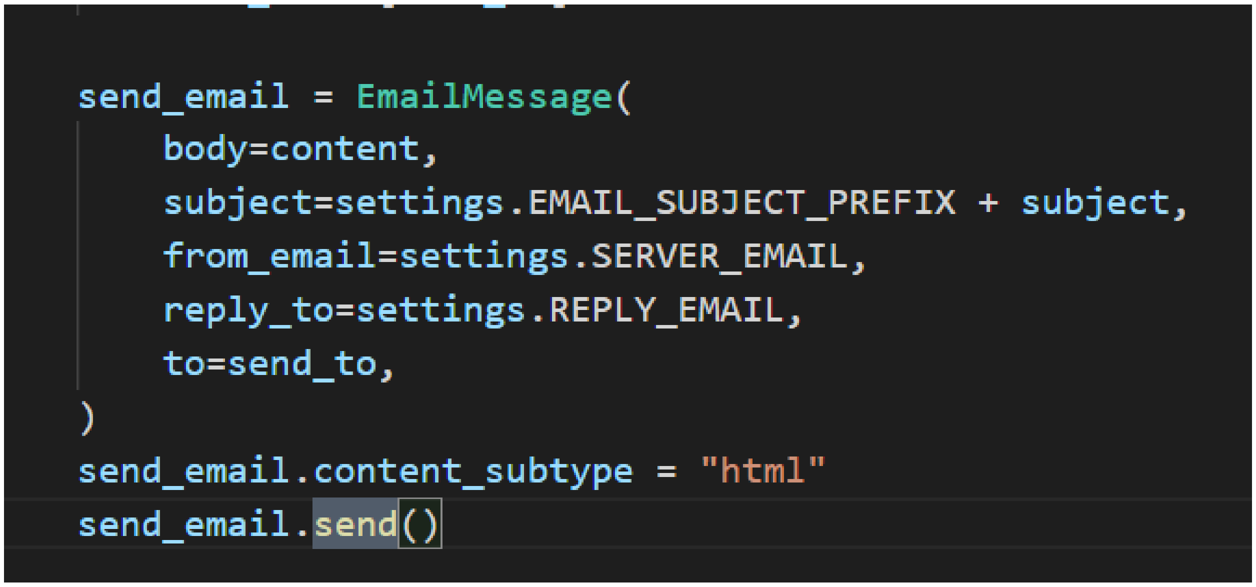
Currently, email is used as a key to fetch Suborg details. The user's id should be used instead of email to fetch the data.



```
@decorators.login_required
def application_list(request):
    applications = SubOrgDetails.objects.filter(suborg_admin_email=request.user.email)
    mentors_list = {}
    for a in applications:
        if hasattr(a.suborg, 'id'):
            mentors_list[a.suborg.id] = UserProfile.objects.filter(role=2, suborg_full_nam
    gsoc_year = GsocYear.objects.first()
    if len(applications) == 0:
        return redirect(reverse("suborg:register_suborg"))
```

4. **Send email page BCC #428 bug**

Currently, the EmailMessage class doesn't contain the bcc. A list of receipt addresses needs to be added to the class on the bcc parameter.

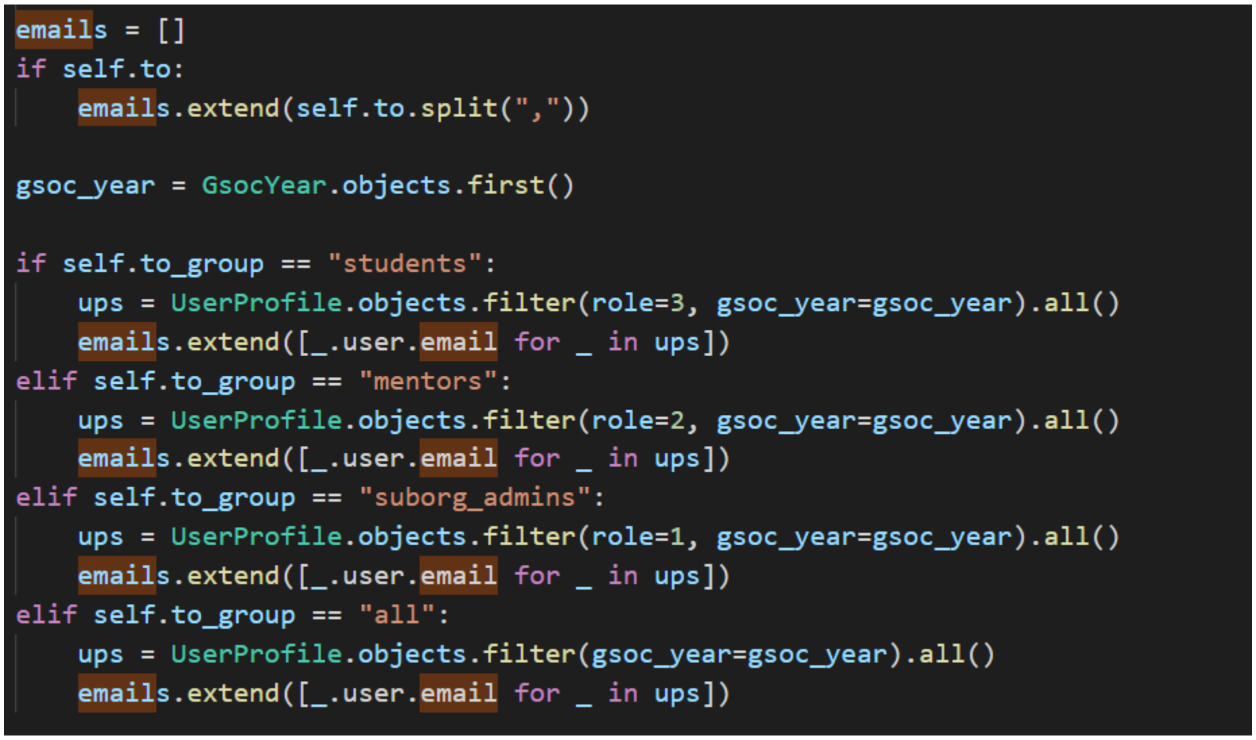


```
send_email = EmailMessage(
    body=content,
    subject=settings.EMAIL_SUBJECT_PREFIX + subject,
    from_email=settings.SERVER_EMAIL,
    reply_to=settings.REPLY_EMAIL,
    to=send_to,
)
send_email.content_subtype = "html"
send_email.send()
```

5. **send email to a group doesn't work #427 bug**

gsoc/models.py line 1133: class SendEmail

Emails are not being added to the list. The email needs to be fetched and added to the list on the following function so that the scheduler doesn't end up sending it to [].



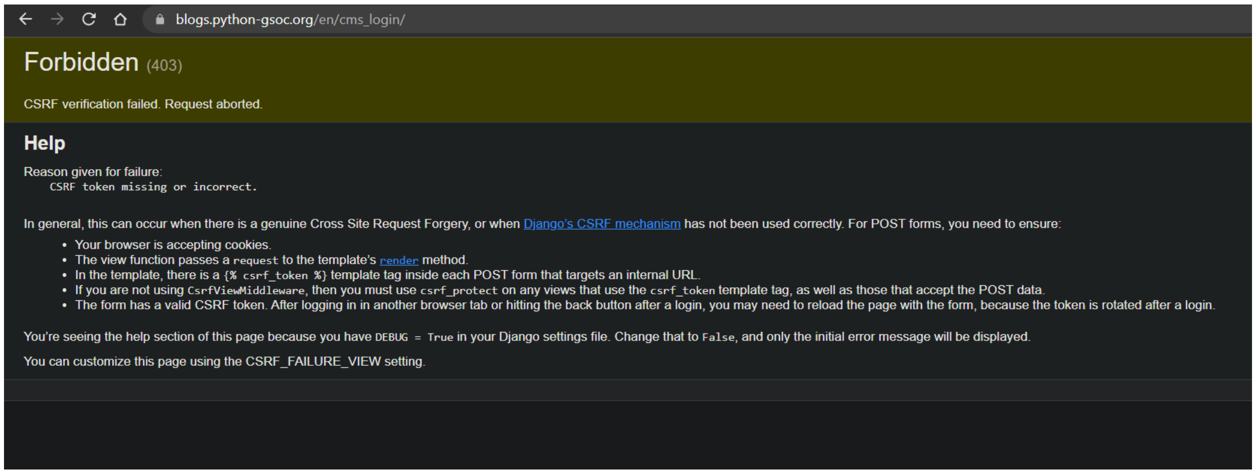
```python
emails = []
if self.to:
    emails.extend(self.to.split(","))

gsoc_year = GsocYear.objects.first()

if self.to_group == "students":
    ups = UserProfile.objects.filter(role=3, gsoc_year=gsoc_year).all()
    emails.extend([_.user.email for _ in ups])
elif self.to_group == "mentors":
    ups = UserProfile.objects.filter(role=2, gsoc_year=gsoc_year).all()
    emails.extend([_.user.email for _ in ups])
elif self.to_group == "suborg_admins":
    ups = UserProfile.objects.filter(role=1, gsoc_year=gsoc_year).all()
    emails.extend([_.user.email for _ in ups])
elif self.to_group == "all":
    ups = UserProfile.objects.filter(gsoc_year=gsoc_year).all()
    emails.extend([_.user.email for _ in ups])
```

6. cannot visit my sub org section #451 bug

This is probably due to CSRF token changes on the browser. Debug needs to be set to False in order to hide such errors or a view can be added to show that CSRF validation has failed.



blogs.python-gsoc.org/en/cms_login/

Forbidden (403)

CSRF verification failed. Request aborted.

Help

Reason given for failure:
```
    CSRF token missing or incorrect.
```

In general, this can occur when there is a genuine Cross Site Request Forgery, or when Django's CSRF mechanism has not been used correctly. For POST forms, you need to ensure:

- Your browser is accepting cookies.
- The view function passes a `request` to the template's `render` method.
- In the template, there is a `{% csrf_token %}` template tag inside each POST form that targets an internal URL.
- If you are not using `CsrfViewMiddleware`, then you must use `csrf_protect` on any views that use the `csrf_token` template tag, as well as those that accept the POST data.
- The form has a valid CSRF token. After logging in in another browser tab or hitting the back button after a login, you may need to reload the page with the form, because the token is rotated after a login.

You're seeing the help section of this page because you have `DEBUG = True` in your Django settings file. Change that to `False`, and only the initial error message will be displayed.

You can customize this page using the CSRF_FAILURE_VIEW setting.

7. **page menu item for students #387 bug**

   DIfferent menu items that are unused for students need to be removed from the toolbar menu when students log in.

8. **occasional duplicate blank user-profiles #455 bug Priority Issue**

   Occasionally some users may end up having duplicate user profiles. This can be eliminated by implementing constraints to restrict duplication.

9. **deleting user-profile when matches #403 bug Priority Issue**

   Currently the Database constraints do not work when the Suborg is blank for users. This leads to having multiple instances of user-profiles. The constraint needs to be deployed to fix that and also user profile deletion should be restricted.

10. **Mentor registering message #460 bug Priority Issue**

    Currently when a mentor is already logged in and clicks on a reglink it says "you have been logged out" the message should be "registered as a mentor with org x please login again".

    The reglink should be checked for if it has been used and an account is created. If it is used then the different dialogue is printed.

    If the email is already registered on the system, the reglink should give a login prompt. This can be achieved by checking if the user is registered.

11. **Unable to make new articles #437 bug Priority Issue**

    Add article button that leads to error needs to be removed.

12. **Add mentor email export (for updating the mailing list) #366 enhancement Priority Issue - Active**

I have already pushed a pull request for this issue. Now the only thing remaining is adding the list to include Suborg admins as well since they are also mentors for projects.

13. **add a link to the list of users by suborg #367 enhancement Priority Issue - Active**

Currently the menu item Users doesn't show the UserProfile list. Instead, it shows the Users list. This aims to fix the issue by showing the UserProfiles based on Suborg from the menu item. The only thing required to do on this issue is to change the title.

14. **Issue #408 - remove SQLite DB and add initial data enhancement**

Currently the project contains an SQLite database file project.db that contains the required initial data for the platform. Instead of having data directly on the database file, this project aims to use Django fixtures to load initial data with ease into the database.

A fixture is a collection of data that Django knows how to import into a database.

First of all, the current demo data need to be dumped into a file. I will use JSON format to dump data as this is the widely used and easy format. The following command is run for all apps on the project.

```
python manage.py dumpdata --format=json appname > data.json
```

The dumped data is saved as following:

```
[
{
        "model": "gsoc.suborg",
        "pk": 1,
        "fields": {
            "suborg_name": "Python Software Foundation GSoC Team"
        }
    },
    {
        "model": "gsoc.suborg",
        "pk": 2,
        "fields": {
            "suborg_name": "CVE Binary Tool"
        }
    }
]
```

After the fixtures are dumped successfully, the data needs to be loaded into the database that will be used with the project.

SQLite is not considered good for the production environment so it is better to use another database system. As stated on issue #388, I will be removing the current SQLite database, integrating a MySQL database, and adding a MySql setup guide.

For MySQL, I will use the mysqlclient python package.
After a successful MySQL setup, the dumped data needs to be loaded into the database in order to be used in the platform.

For loading the data from the fixtures, I will use the following command:

```
python manage.py loaddata data.json
```

15. **Issue #385 - Add a view to see what students for the current year haven't accepted the invite enhancement Highest Priority**

For this issue, the invited students need to be checked for if they have registered into the system or not. Data under the RegLink model needs to be retrieved and compared against the current system users to determine if they have accepted the invite successfully or not.

Weekly Timeline:

I. **Community Bonding** (May 20 - June 12)
   - Get familiarized with the mentors and the admins of the project and discuss the vision of the project and gain insight about implementing the enhancements.

II. **Week 1** (June 13 - June 19)
   - Start working on Issue #453
   - Implement the fixes for issue #453
   - Start working on Issue #457
   - Implement fixes for issue #457

III. **Week 2** (June 20 - June 27)
   - Start working on Issue #433
   - Implement the fixes for issue #433
   - Start working on Issue #428
   - Implement fixes for issue #428

IV. **Week 3** (June 28 - July 5)
   - Start working on Issue #427
   - Implement the fixes for issue #427
   - Start working on Issue #451
   - Implement fixes for issue #451

V. **Week 4** (July 6 - July 13)
   - Start working on Issue #387
   - Implement the fixes for issue #387
   - Start working on Issue #455

- Implement fixes for issue #455

VI. **Week 5** (July 14 - July 21)
  - Start working on Issue #403
  - Implement the fixes for issue #403
  - Start working on Issue #460
  - Implement fixes for issue #460

VII. **Week 6** (July 22 - July 29)
  - Start working on Issue #437
  - Implement the fixes for issue #437
  - Start working on Issue #366
  - Implement fixes for issue #366

VIII. **Week 7** (July 30 - August 6)
  - Start working on Issue #408
  - Implement fixes for issue #408

IX. **Week 8** (August 7 - August 14)
  - Start working on Issue #367
  - Implement the fixes for issue #367
  - Start working on Issue #385
  - Implement the fixes for issue #385

X. **Week 9 - 11** (August 15 - September 4)
  - Start working on other issues listed on the issue tracker
  - Discuss with mentors regarding the issue
  - Implement the fixes

XI. **Final Week** (September 5 - September 12)

- Refactoring and cleaning up code
- Fixing additional bugs that pop up on the portal
- Updating documentation if any changes are required

Other Commitments

I will have my classes except for Saturday or any other public holidays. I will be able to manage 4-5 hours on the days I will have classes. I will be committing more time on Saturday and other holidays.

This is my first time applying to the GSoC and I do not plan to apply to any other organization.