API used in the module

There are no actual APIs implemented here in this module.

Notifications are dispatched by invoking functions within the views of the notification module from various modules. Each module triggers the function by passing the necessary parameters, resulting in the delivery of the corresponding message. The URLs have been hard coded and requests are being rendered

Overview of the module:-

Notify the User about relevant events, such as interactions with their content and updates from others students and faculty. For every module there will be different type of notifications according to the unique needs and activities associated with each individual module.

- View notification (implemented in all the modules)
- Delete notification (not implemented)
- Make announcement
- Mark as read(implemented in all the modules)
- Mark as unread(implemented in all the modules)

1. File tracking module

```
def file_tracking_notif(sender, recipient,title):
    url='filetracking:inward'
    module='File Tracking'
    sender = sender
    recipient = recipient
    verb = title

            notify.send(sender=sender,     recipient=recipient,     url=url,
module=module, verb=verb)
```

```
file_tracking_notif(request.user,receiver_id,subject)
                messages.success(request,'File sent successfully')
```

Explanation: In the file tracking module, upon sending a notification, the sender will receive an acknowledgment notification confirming that the file has been sent successfully. Simultaneously, the recipient will be notified that the file has been received.

2.Health center module

```
def healthcare_center_notif(sender, recipient, type):
    url='healthcenter:healthcenter'
    module='Healthcare Center'
    sender = sender
    recipient = recipient
    verb = ''
    if type == 'appoint':
        verb = "Your Appointment has been booked"
    if type == 'amb_request':
        verb = "Your Ambulance request has been placed"
    if type == 'Presc':
        verb = "You have been prescribed some medicine"
    if type == 'appoint_req':
        verb = "You have a new appointment request"
    if type == 'amb_req':
        verb = "You have a new ambulance request"
 notify.send(sender=sender, recipient=recipient, url=url, module=module,
verb=verb)
```

Explanation: The doctor appointment form within the **health center module** is not working, while the ambulance request function is operational, and ambulance requests are directed to the compounder. After submitting feedback, it is unclear who receives the notifications. Following the argument passage, the function sends a specific message corresponding to the provided type.

3.Visitors hostel module

```
def visitors_hostel_notif(sender, recipient, type):
    url='visitorhostel:visitorhostel'
    module="Visitor's Hostel"
    sender = sender
    recipient = recipient
    verb = ''
    if type =='booking_confirmation':
        verb='Your booking has been confirmed '
    elif type =='booking_cancellation_request_accepted':
        verb='Your Booking Cancellation Request has been accepted '
    elif type =='booking_request':
```

```
        verb='New Booking Request '
    elif type =='cancellation_request_placed':
        verb='New Booking Cancellation Request '
    elif type =='booking_forwarded':
        verb='New Forwarded Booking Request '
    elif type =='booking_rejected':
        verb='Your Booking Request has been rejected '

          notify.send(sender=sender,    recipient=recipient,    url=url,
module=module, verb=verb)
```

```
visitors_hostel_notif(request.user, bd.intender, 'booking_confirmation')
        return HttpResponseRedirect('/visitorhostel/')
    else:
        return HttpResponseRedirect('/visitorhostel/')
```

Explanation: Notifications are not functioning in the **Visitors hostel module** due to its incomplete functionality. The form details are not being transmitted to the caretaker responsible for the visitors' hostel.

4.Gymkhana module

GYMKHANA_VOTING

```
def gymkhana_voting(sender, recipient, type, title, desc):
  url = 'gymkhana:gymkhana'
  module = 'Gymkhana Module'
  sender = sender
  recipient = recipient
  title = title
  desc = desc
  verb = ""
  if type == 'voting_open':
      verb = "Voting is open for {}".format(title)

  notify.send(sender=sender,
              recipient=recipient,
              url=url,
              module=module,
              verb=verb,
              description=desc
              )
```

```
```

GYMKHANA_SESSION

```
     def gymkhana_session(sender, recipient, type, club, desc, venue):
    url = 'gymkhana:gymkhana'
    module = 'Gymkhana Module'
    sender = sender
    recipient = recipient
    desc = desc
    verb = ""
    if type == 'new_session':
       verb = "A session by {} Club will be organised in {}".format(club,
venue)

    notify.send(sender=sender,
                recipient=recipient,
                url=url,
                module=module,
                verb=verb,
                description=desc
                )
```

GYMKHANA_EVENT

```
def  gymkhana_event(sender,  recipient,  type,  club,  event_name,  desc,
venue):
    url = 'gymkhana:gymkhana'
    module = 'Gymkhana Module'
    sender = sender
    recipient = recipient
    desc = desc
    verb = ""

   if type == 'new_event':
     verb    =    "{}    event    by    {}    Club    will    be    organised
in{}".format(event_name,club,venue)

    notify.send(sender=sender,
                recipient=recipient,
                url=url,
                module=module,
                verb=verb,
                description=desc
    )
```

Within the **Gymkhana module**, all notifications are operational, except for the club membership form, where the selection of the club is not functioning, rendering it inactive. The co-convener can send voting poll notifications to specific batches, event notifications to all batches, and session notifications to all batches within the gymkhana module.

The notifications mentioned earlier can be directed to either an individual or a group, such as the entire batch of 2021.e

5.Research procedures module

In the **research procedures module**, the patent form is not working, so it is currently non-functional in generating notifications. Similarly, within the other academic procedures module, both the leave module and assistant claimship module forms have notification functions defined, but they are not working. These are currently not active for generating notifications

6.Complaint system module

In the **complaint system module**, once a student submits a complaint, a notification will be forwarded to the designated caretaker assigned to handle that specific complaint.

7.Mess module

Within the **mess module**, notifications will be accessible upon submission of feedback, requests for changes in the mess menu, and leave or vacation requests from the staff. Additionally, when a student is added to the mess committee, they will receive a notification informing them of their new role.

8.Department module

In **Department module** the publication of an announcement by the department head, notifications will be sent to both the department's students and the head of the department. These notifications will be accessible in the announcements section.

```
urlpatterns = [
                            pattern(r'^mark-as-read-and-redirect/(?P<slug>\d+)/$',
views.mark_as_read_and_redirect, name='mark_as_read_and_redirect'),
    ] + urlpatterns
```

mark-as-read-and-redirect/()/ this URL marks a notification as read and redirects the user based on the notification's data.

```
def mark_as_read_and_redirect(request, slug=None):
    notification_id = slug2id(slug)
    notification = get_object_or_404(
        Notification, recipient=request.user, id=notification_id)
    notification.mark_as_read()
    if(notification.data['module'] == 'Complaint System'):
        complaint_id=notification.description
                                                                    return
HttpResponseRedirect(reverse(notification.data['url'],kwargs={'detailcomp_id1':
complaint_id}))
    else:
        return HttpResponseRedirect(reverse(notification.data['url']))
```

This URL uses make use of the mark_as_read_and_redirect function

If the notification is related to the "Complaint System" module, it redirects the user to a specific URL with additional parameters. Otherwise, it redirects the user to a general URL provided in the notification's data.

How the functions in the views works..

The notification takes sender id and recipient id and title as parameters and

sends a notification using the notify. send function. The notification content is determined based on the type parameter, and it includes information about the sender, recipient, URL, and module. The above function was invoked within that particular module file, and it forwarded the arguments that the function needs to utilize in order to dispatch the notification.

```
urlpatterns = [
                                    pattern(r'^url_specification/(?P<slug>\d+)/$',
views.mark_as_read_and_redirect, name=function_name'),
    ] + urlpatterns
```

Current problems you are facing with the module or in its use cases —

- **View notification**: Determining the recipient's credentials is proving challenging, making it somewhat difficult to verify the successful delivery of notifications.
- **delete notification**: This Use Case is not currently implemented

  https://docs.google.com/document/d/1CF0igjB9dvkaMYXIv9XpLwzMMqJE5Dy5-xpuwIuTmHA/edit?usp=sharing