



# Research Match Sprint 7

Deadline Tech



# Unique Profile Page

```
path('profile/<str:pk>', views.profile, name='profile')
```

```
def profile(request, pk):
    pk += '@emory.edu'
    user_object = User.objects.get(username=pk)
    user_profile = StudentProfile.objects.get(user=user_object)

    context = {
        'user_object': user_object,
        'user_profile': user_profile,
    }

    if user_profile.is_student:
        return render(request, 'StudentMain.html', context)
    elif user_profile.is_lab:
        return render(request, 'LabMain.html', context)
    else:
        return render(request, 'home.html', context)
```

- Each user has their own profile page. Students can look at lab profile pages.

# A Student's matched Labs

## Matched Labs



**Dooley's CS Lab**

### Skills Required

python

### Courses Required

CS 325 CS 377


# Lab's matched students

← → ↺ 127.0.0.1:8000/matchedstudents/ygutierrezyadi

Research Match

Profile Matched Students Students Settings

## Matched Students



**Yareli Gutierrez**  
yareli.gutierrez@emory.edu

**Skills**

java

**Courses**

CS 326

```
matches = models.ManyToManyField("self", related_name='matched_by', symmetrical=False, blank=True)
```

```
@allowed_users(allowed_roles=['student'])
def matches(request, pk):
    pk += '@emory.edu'
    user_object = User.objects.get(username=pk)
    user_profile = StudentProfile.objects.get(user=user_object)

    context = {
        'user_object': user_object,
        'user_profile': user_profile,
    }
    return render(request, 'matches.html', context)
```

```
{% comment %} This for loop lists all users that match with the current user. {% endcomment %}
{% for profile in user_profile.matches.all %}
```

```
#These functions get the user's information from Django's User model.
def get_user_name(self):
    return self.user.email.split('@')[0]
def get_first_name(self):
    return self.user.first_name
def get_last_name(self):
    return self.user.last_name
def get_email(self):
    return self.user.email
```

- This allows the student user to view all matched labs and vice versa.

# Modified Form

## Update Your Profile

First name:

Last name:

Profile pic: Currently: [profile\\_pics/69544005426\\_CF472075-4C3C-443C-B0AE-19615F64F1DD.jpeg](profile_pics/69544005426_CF472075-4C3C-443C-B0AE-19615F64F1DD.jpeg)

Change:  No file chosen

Skill:

Course:

Biography:

Documents: Currently: [documents/Arushi\\_Dhillon\\_Transcript.pdf](documents/Arushi_Dhillon_Transcript.pdf)

Change:  No file chosen

## Update Your Profile

First name:

Last name:

Profile pic: Currently: [profile\\_pics/DSC00011.jpeg](profile_pics/DSC00011.jpeg)

Change:  No file chosen

Skill:  Separate skills with a comma.

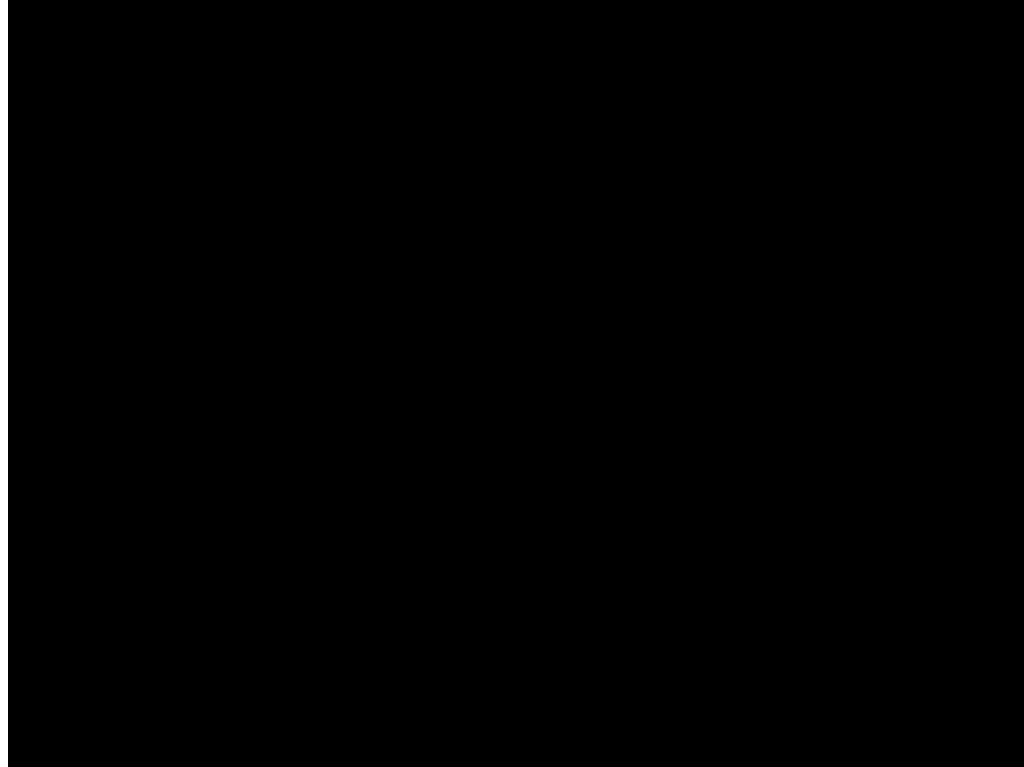
Course:  Separate courses with a comma.

Biography:

Documents: Currently: [documents/Procedure\\_Physiccs.docx](documents/Procedure_Physiccs.docx)

Change:  No file chosen

# Downloadable Files



# Search

```
def search(request):
    return render(request, 'search.html')

def search_profiles(request):
    if request.htmx: # Check for the HTMX request header
        query = request.GET.get('q', '')

        if len(query) > 0:
            # Search profiles by matching the query with first name, last name, or background fields
            profiles = StudentProfile.objects.filter(
                Q(firstname__icontains=query) |
                Q(lastname__icontains=query) |
                Q(background__icontains=query) |
                Q(labname__icontains=query)
            )

            # Render only the search results to a specific template for HTMX to swap in
            return render(request, 'search/list_profiles.html', {'profiles': profiles})
        else:
            # If query is empty, return an empty response to clear the results
            return HttpResponse('')

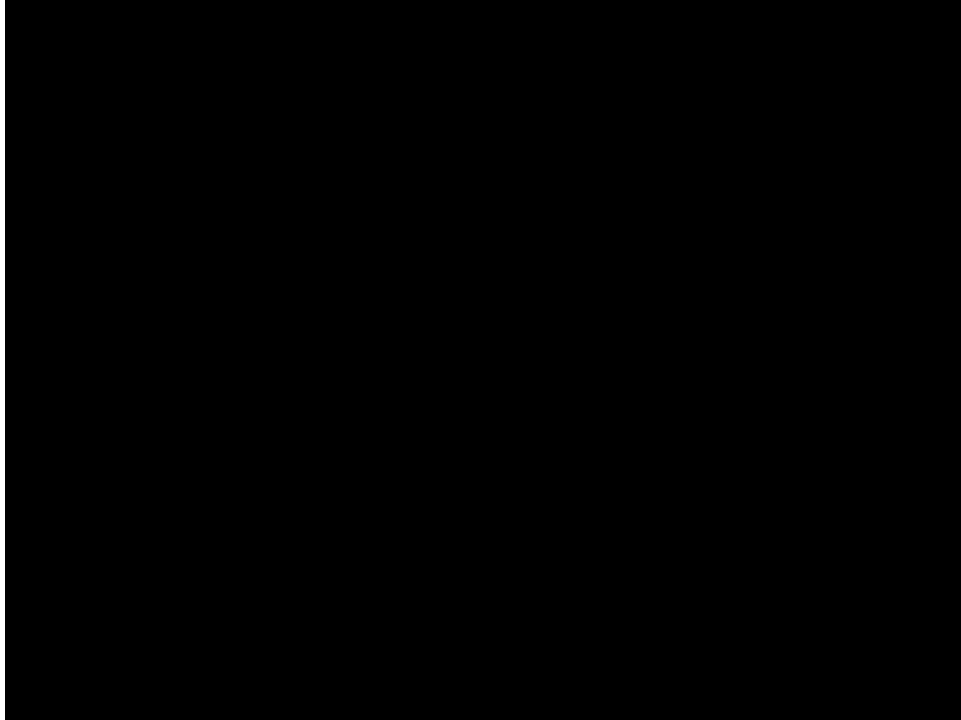
    else:
        # If it's not an HTMX request, raise a 404
        raise Http404('')

#Function to display the details of the profile
def profile_detail(request, id):
    StudentProfile = get_object_or_404(StudentProfile, id=id)

    #Check the type of profile and redirect it accordingly
    if StudentProfile.StudentProfile_BACKGROUND == "Student":
        return render(request, 'profilepage/StudentMain.html', {'StudentProfile': StudentProfile})
    elif StudentProfile.StudentProfile_BACKGROUND == "Mentor":
        return render(request, 'profilepage/LabMain.html', {'StudentProfile': StudentProfile})
    else:
        raise Http404()
```



# Search



# Increasing Security with Secure Socket Layer (SSL)

- Added SSL certificate to server for secure and encrypted connections
- Configured email verification to use custom domains as gmail is rejecting current authentication request

```
# Email configuration using SendGrid as the email backend
EMAIL_BACKEND = 'django.core.mail.backends.smtp.EmailBackend'

# Get SendGrid API key from environment variables
SENDGRID_API_KEY = os.getenv('SENDGRID_API_KEY')

# Use TLS (Transport Layer Security) for secure email communication
EMAIL_USE_TLS = True

# SendGrid SMTP server address and api name
EMAIL_HOST = 'smtp.sendgrid.net'
EMAIL_HOST_USER = 'apikey'
# SendGrid SMTP server password (using the SendGrid API key)
EMAIL_HOST_PASSWORD = SENDGRID_API_KEY
# SendGrid SMTP server port
EMAIL_PORT = 587
# Enable email debugging (set to True for debugging purposes)
EMAIL_DEBUG = True
# Redirect all HTTP requests to HTTPS for secure communication
SECURE_SSL_REDIRECT = True
# Define the header to use for indicating the original protocol when behind a proxy (Heroku)
SECURE_PROXY_SSL_HEADER = ('HTTP_X_FORWARDED_PROTO', 'https')
```

# Inbox

## New Message

To:

 [bob jones](#)

 [joe smith](#)



**bob jones**

bob

Adrian

Nov. 15, 2023, 6:58 p.m. 14 minutes

hey bob!

Adrian

Nov. 15, 2023, 7:09 p.m. 3 minutes

do you have an open lab position?

New Message

```
16 <body>
17 <div class="wrapper">
18   <inbox class="inbox">
19     <div class="inboxleft">
20       {% include 'form_searchuser.html' %}
21       {% include 'my_conversations.html' %}
22       <div id="open-conversations"></div>
23     </div>
24     <div class="inboxright">
25       {% include 'conversation.html' %}
26     </div>
27   </inbox>
28 </div>
29 </body>
```





# Inbox - Reply

## New Message

To:



[joe.smith](#)

[bob.jones](#)



**joe smith**

joe

Adrian

Nov. 15, 2023, 6:58 p.m. 58 minutes

hey joe! can i join your lab?

Adrian

Nov. 15, 2023, 7:24 p.m. 33 minutes

hey



Add message ...

Send Message

[Cancel](#)

```
1 <div >
2   
3   <form method='POST' action="{ url 'inbox-newreply' conversation.id %}">
4     {% csrf_token %}
5     {{ new_message_form }}
6     <div>
7       <button type="submit">Send Message</button>
8       <a href="{ request.META.HTTP_REFERER }">Cancel</a>
9     </div>
10  </form>
11 </div>
```

```
6 class InboxNewMessageForm(ModelForm):
7     class Meta:
8         model = InboxMessage
9         fields = ['body']
10        labels = {
11            'body' : '',
12        }
13        widgets = {
14            'body' : forms.Textarea(attrs={'rows': 4, 'placeholder': 'Add message ...'})
15        }
```

# Inbox - Messages

- Reply

```
<div>
  <a
    hx-get="{% url 'inbox-newreply' conversation.id %}"
    hx-target="this"
    hx-swap="outerHTML scroll:#conversation:bottom" >
    <div class="btn">New Message</div>
  </a>
</div>
```

- No conversation selected

```
52  {% else %}
53  <div>
54    |   Select a conversation or create a new message
55  </div>
56  {% endif %}
57  </div>
```

# Inbox - Search

```
1 <h3 id="modal-title">
2   New Message
3 </h3>
4 <div id="new-message">
5   <div>
6     <span>To:</span>
7     <input placeholder="Search user ..." type="text" name="search_user"
8       hx-get="{% url 'inbox-searchusers' %}"
9       hx-trigger="keyup changed"
10      hx-target="#search-results"
11      hx-swap="innerHTML" >
12   </div>
13   <div id="search-results">
14   </div>
15 </div>
```



# Inbox - Search

## New Message

To:



Jimmy Jim jimmy



[joe smith](#)



[bob jones](#)

```
1 <ul id="user-list">
2   {% for user in users %}
3   <li>
4     <a
5       hx-get="{% url 'inbox-newmessage' user.id %}"
6       hx-target="#conversation"
7       hx-swap="innerHTML"
8     >
9       
10      <div class="btn">
11        {{ user.first_name }} {{ user.last_name }}
12        <span>{{ user.username }}</span>
13      </div>
14    </a>
15  </li>
16  {% endfor %}
17 </ul>
```

# Inbox - Search

```
37 def search_users(request):
38     if request.htmx:
39         letters = request.GET.get('search_user')
40         if len(letters) > 0:
41             profiles = User.objects.filter(first_name__contains=letters).exclude(username=request.user)
42             return render(request, 'list_searchuser.html', { 'users' : profiles })
43         else:
44             return HttpResponse('')
45     else:
46         raise Http404()
```

# Inbox - New Message

## New Message

To:



Jimmy Jim jimmy

[joe smith](#)

[bob jones](#)

To:



jimmy

Add message ...

[Cancel](#)

```
1 <div>
2   <span>To:</span>
3   <div>
4     
5     <div>
6       {{ recipient.studentprofile.first_name }}
7       <span>{{ recipient.username }}</span>
8     </div>
9   </div>
10 </div>
11 </div>
12
13 <div class="newmessage">
14   <form method="POST" action="{% url 'inbox-newmessage' recipient.id %}">
15     {% csrf_token %} {{ new_message_form }}
16     <button type="submit">Submit Message</button>
17     <a href="{{ request.META.HTTP_REFERER }}">Cancel</a>
18   </form>
19 </div>
20
```



Thank you!

