



Research Match: Sprint 3

Deadline Tech



Feedback

- Creating a difference between lab page and student page.
- Is the current labs page necessary? Considering removing it or moving it to the end of our priority list.
- Shorter summaries and information on the opportunities page- additional information after you click on labs/students.
- Actively looking for students or not option for labs.
- Connect with professors in different areas of study.

Feedback

Research Match

☆

👤

Board

▼

🔍

⚡

☰

Filters

AD

AV

DL

GD

+2

Share

...

Product Backlog

...

Create example dataset for backend

Matching algorithm

Responsive layout for mobile and wide screens

AV

Create database for courses.

Write algorithm to match student skills with professor requirements.

Write and algorithm to match student courses with professor requirements.

Store and display student biography.

Shorter summaries and info on the opportunities page, more info after clicking on options

actively searching for researchers option to opt out of matching

+ Add a card

📄

To Do

...

Organize GitHub repo/move html files to django folders

AV

Display skills from student profile to the student page that the lab will see.

●

Store Email information

●

YG

+ Add a card

📄

Doing

...

Login Page/Sign in page

☑ 0/5

YG

Flesh out admin panel

GD

Host database and webpages through Heroku

GD

Make add CSV function to populate database

GD

Student Profile Page

☑ 0/20

●

Lab Profile Page

☰

AV

Create Match Page

AD

+ Add a card

📄

Done

...

Opportunities Page

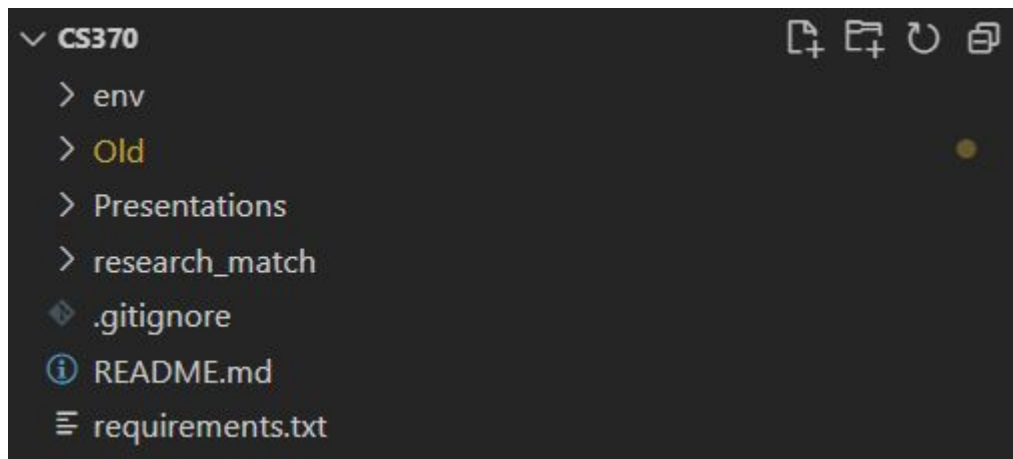
Add navbar to all pages

+ Add a card

📄

Front End

- Migrated front-end to Django
- Organized repository
- Settings page





Demo



Inbox Function - Back End

```
class InboxMessage(models.Model):
    sender = models.ForeignKey(User, on_delete=models.CASCADE, related_name="sent_messages")
    conversation = models.ForeignKey('Conversation', on_delete=models.CASCADE, related_name="messages")
    body = models.TextField()
    created = models.DateTimeField(auto_now_add=True)

    @property
    def body_decrypted(self):
        f = Fernet(settings.ENCRYPT_KEY)
        message_decrypted = f.decrypt(self.body)
        message_decoded = message_decrypted.decode('utf-8')
        return message_decoded

class Meta:
    ordering = ['-created']

    def __str__(self):
        time_since = timesince(self.created, timezone.now())
        return f'[{self.sender.username} : {time_since} ago]'

class Conversation(models.Model):
    id = models.CharField(max_length=100, default=uuid.uuid4, unique=True, primary_key=True, editable=False)
    participants = models.ManyToManyField(User, related_name='conversations')
    lastmessage_created = models.DateTimeField(default=timezone.now)
    is_seen = models.BooleanField(default=False)

    class Meta:
        ordering = ['-lastmessage_created']

    def __str__(self):
        user_names = ", ".join(user.username for user in self.participants.all())
        return f'[{user_names}]'
```

```
@login_required
def inbox_view(request, conversation_id=None):
    my_conversations = Conversation.objects.filter(participants=request.user)
    if conversation_id:
        conversation = get_object_or_404(my_conversations, id=conversation_id)
        latest_message = conversation.messages.first()
        if conversation.is_seen == False and latest_message.sender != request.user:
            conversation.is_seen = True
            conversation.save()
    else:
        conversation = None
    context = {
        'conversation': conversation,
        'my_conversations': my_conversations
    }
    return render(request, 'inbox/inbox.html', context)

def search_users(request):
    if request.htmx:
        letters = request.GET.get('search_user')
        if len(letters) > 0:
            profiles = Profile.objects.filter(realname__icontains=letters)
            users_id = profiles.values_list('user', flat=True)
            users = User.objects.filter(
                Q(username__icontains=letters) | Q(id__in=users_id)
            ).exclude(username=request.user.username)
            return render(request, 'inbox/list_searchuser.html', {'users': users})
        else:
            return HttpResponse('')
    else:
        raise Http404()
```

```

@login_required
def new_reply(request, conversation_id):
    new_message_form = InboxNewMessageForm()
    my_conversations = request.user.conversations.all()
    conversation = get_object_or_404(my_conversations, id=conversation_id)

    if request.method == 'POST':
        form = InboxNewMessageForm(request.POST)
        if form.is_valid():
            message = form.save(commit=False)

            # encrypt message
            message_original = form.cleaned_data['body']
            message_bytes = message_original.encode('utf-8')
            message_encrypted = f.encrypt(message_bytes)
            message_decoded = message_encrypted.decode('utf-8')
            message.body = message_decoded

            message.sender = request.user
            message.conversation = conversation
            message.save()
            conversation.lastmessage_created = timezone.now()
            conversation.is_seen = False
            conversation.save()
            return redirect('inbox', conversation.id)

    context = {
        'new_message_form': new_message_form,
        'conversation': conversation
    }
    return render(request, 'inbox/form_newreply.html', context)

def notify_newmessage(request, conversation_id):
    conversation = get_object_or_404(Conversation, id=conversation_id)
    latest_message = conversation.messages.first()
    if conversation.is_seen == False and latest_message.sender != request.user:
        return render(request, 'a_inbox/notify_icon.html')
    else:
        return HttpResponse('')

def notify_inbox(request):
    my_conversations = Conversation.objects.filter(participants=request.user, is_seen=False)
    for c in my_conversations:
        latest_message = c.messages.first()
        if latest_message.sender != request.user:
            return render(request, 'inbox/notify_icon.html')
    return HttpResponse('')

```

```

@login_required
def new_message(request, recipient_id):
    recipient = get_object_or_404(User, id=recipient_id)
    new_message_form = InboxNewMessageForm()

    if request.method == 'POST':
        form = InboxNewMessageForm(request.POST)
        if form.is_valid():
            message = form.save(commit=False)

            # encrypt message
            message_original = form.cleaned_data['body']
            message_bytes = message_original.encode('utf-8')
            message_encrypted = f.encrypt(message_bytes)
            message_decoded = message_encrypted.decode('utf-8')
            message.body = message_decoded

            message.sender = request.user

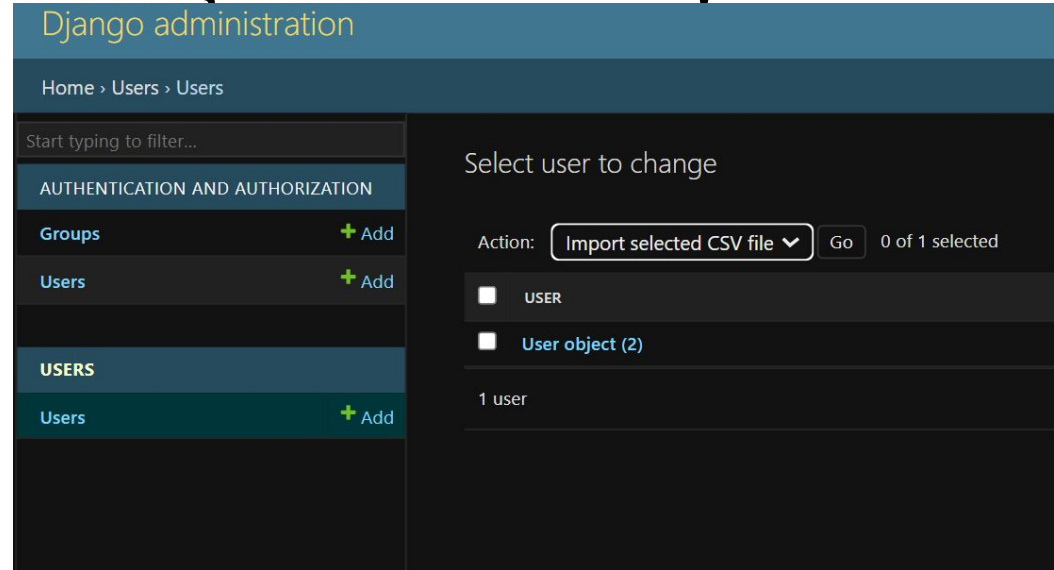
            my_conversations = request.user.conversations.all()
            for c in my_conversations:
                if recipient in c.participants.all():
                    message.conversation = c
                    message.save()
                    c.lastmessage_created = timezone.now()
                    c.is_seen = False
                    c.save()
                    return redirect('inbox', c.id)
            new_conversation = Conversation.objects.create()
            new_conversation.participants.add(request.user, recipient)
            new_conversation.save()
            message.conversation = new_conversation
            message.save()
            return redirect('inbox', new_conversation.id)

    context = {
        'recipient': recipient,
        'new_message_form': new_message_form
    }
    return render(request, 'inbox/form_newmessage.html', context)

```

Building Admin Page Functionality

- Generate student test data for initial testing and bug fixing
- Build Admin page for easy data management
- Develop CSV import function for fast upload and management of test data



username	email	firstname	lastname	background	subject
mmcadam0	mmcadam0@bandcamp.com	Marchelle	McAdam	M	MATH
sjaze1	sjaze1@unblog.fr	Saundra	Jaze	M	MATH
msaintepaul2	msaintepaul2@yahoo.com	Malia	Sainte Paul	M	BIOL
sbarrow3	sbarrow3@alexa.com	Starla	Barrow	M	BIOL
ybratcher4	ybratcher4@bluehost.com	Yuri	Bratcher	S	MATH

Building Admin Page Functionality

```
class MyModelAdmin(admin.ModelAdmin):
    actions = ['import_csv']

    def import_csv(self, request, queryset):
        if 'input_file' in request.FILES:
            csv_file = request.FILES['input_file']
            decoded_file = csv_file.read().decode('utf-8').splitlines()

            for row in csv.DictReader(decoded_file):
                User.objects.update_or_create(
                    username=row['username'],
                    defaults={'email': row['email'], 'firstname': row['firstname'], 'lastname': row['lastname'],
                              'background': row['background'], 'subject': row['subject']},
                )

            self.message_user(request, "CSV file imported successfully.")
        else:
            self.message_user(request, "No CSV file provided.")

import_csv.short_description = "Import selected CSV file"
```

Building Admin Page Functionality

```
def get_actions(self, request):
    actions = super().get_actions(request)
    if 'delete_selected' in actions:
        del actions['delete_selected']
    return actions

def import_csv_form(self):
    return mark_safe(
        """
        <form method="post" enctype="multipart/form-data">
            {% csrf_token %}
            <p>Upload a CSV file:</p>
            <input type="file" name="input_file" accept=".csv">
            <input type="submit" value="Upload and Update">
        </form>
        """
    )

def changelist_view(self, request, extra_context=None):
    extra_context = extra_context or {}
    extra_context['import_csv_form'] = self.import_csv_form()
    return super().changelist_view(request, extra_context=extra_context)
```

```
admin.site.register(User, MyModelAdmin)
```

Sign Up / Log In Page

```
def signup(request):  
    if request.method == "POST":  
        firstname = request.POST['firstname']  
        lastname = request.POST['lastname']  
        email = request.POST['email']  
        pass1 = request.POST['pass1']  
        pass2 = request.POST['pass2']  
  
        if User.objects.filter(email=email):  
            messages.error(request, "Email already exists! Please try some other email")  
            return redirect('home')  
  
        if pass1 != pass2:  
            messages.error(request, "Passwords didn't match!")  
  
        myuser = User.objects.create_user(email, email, pass1)  
        myuser.first_name = firstname  
        myuser.last_name = lastname  
        myuser.is_active = False  
  
        myuser.save()
```

- Connect signup/login page with Django email verification

```
# Welcome Email
subject = "Welcome to Research Match Login!"
message = "Hello " + myuser.first_name + "!! \n" + "Welcome to Research Match"
from_email = settings.EMAIL_HOST_USER
to_list = [myuser.email]
send_mail(subject, message, from_email, to_list, fail_silently=True)
```

```
# Email Address Confirmation Email
```

```
current_site = get_current_site(request)
email_subject = "Confirm your email @ Research Match Login!"
message2 = render_to_string('email_confirmation.html',{

    'name': myuser.first_name,
    'domain': current_site.domain,
    'uid': urlsafe_base64_encode(force_bytes(myuser.pk)),
    'token': generate_token.make_token(myuser)
})
```

```
email = EmailMessage(
    email_subject,
    message2,
    settings.EMAIL_HOST_USER,
    [myuser.email],
)


email.fail_silently = True
email.send()

return redirect('studentlogin')
```

[External] Welcome to Research Match Login!



researchmatchdemo@gmail.com

To  Gutierrez, Yareli

Retention Policy Junk Email (30 days)

Expires 11/17/2023



This item will expire in 30 days. To keep this item longer apply a different Retention Policy. Links and other functionality have been disabled in this message. To turn on that functionality, move this message to the Inbox.

LinkedIn

Hello yareli!!

Welcome to Research Match!

Thank you for visiting our website

We have also sent you a confirmation email, please confirm your email address in order to activate your account.

Thanking you

Deadline Tech

```
from django.contrib.auth.tokens import PasswordResetTokenGenerator
from six import text_type

class TokenGenerator(PasswordResetTokenGenerator):
    def _make_hash_value(self, user, timestamp):
        return (
            text_type(user.pk) + text_type(timestamp)
        )
```

```
generate_token = TokenGenerator()
```

```
def activate(request, uidb64, token):
    try:
        uid = force_str(urlsafe_base64_decode(uidb64))
        myuser = User.objects.get(pk=uid)

    except(TypeError, ValueError, OverflowError, User.DoesNotExist):
        myuser = None

    if myuser is not None and generate_token.check_token(myuser, token):
        myuser.is_active = True
        myuser.save()
        login(request, myuser)
        return redirect('home')

    else:
        return render(request, 'activation_failed.html')
```

```
def studentlogin(request):

    if request.method == 'POST':
        email = request.POST['email']
        password = request.POST['pass1']

        user = authenticate(username=email, password=password)

        if user is not None:
            login(request, user)
            firstname = user.first_name
            return render(request, "profile.html", {'firstname': firstname})

        else:
            messages.error(request, "Bad Credentials!")
            return redirect('home')

    return render(request, "registration/loginpage.html")
```

Research Match

Login or Signup to access Research Match

Student Lab **Signup**

Enter Your First Name

Enter Your Last Name

email@emory.edu

Create a Password

Confirm Your Password

**Signup as a
Student**

Signup as a Lab

Next Sprint

- Use encryption for passwords of users instead of storing actual passwords
- Connect Emory smtp server
- Connect Login Page with other RM pages such as Student Profile/Lab Profile
- Reset Password Page is created; I will code the reset password function through the generation token and email verification function in Django.

Display User Input to Other Locations

```
def get_skill(request):
    # if this is a POST request we need to process the form data
    if request.method == "POST":
        # create a form instance and populate it with data from the request
        form = SkillForm(request.POST)
        # check whether it's valid:
        # if form.is_valid():
            # process the data in form.cleaned_data as required
            # ...
            # redirect to a new URL:
            skill_input = form.cleaned_data['skill']
            p = Skills(skill_input)
            p.save()

            # return HttpResponseRedirect("/thanks/")

    # if a GET (or any other method) we'll create a blank form
    else:
        form = SkillForm()

    return render(request, "StudentMain.html", {"form": form})
```

```
<div class="Identity">
    
    <div class="Identityinfo">
        <h2 class="StudentName">Eli Dhillon</h2>
        <div class="tags">
            <span class="tag"> GPA: 3.9</span>
            <!-- <div id="Skill_item">
            </div> -->
            <div>
                {% for x in Skills %}
                <tr>
                    <td>{{ x.skill }}</td>
                </tr>
                {% endfor %}
            </div>
        </div>
    </div>
    <form action="" method="POST">
        <label for="Skill_input">Skills: </label>
        <!-- <input type='text' id='Skill_input' /> -->
        {% csrf_token %}
```



Thank You!

