Compserv Onboarding

Fall 2025

CompServ

CompServ is in charge of computing services, member logistics, and managing the website for HKN.

Currently our biggest project is the overhaul of the website @ **dev-hkn.eecs.berkeley.edu**.

Committee Project:

- Attend compserv events!
- Meet the compserv officers!
- Contribute to our new HKN website!
  - Add features
  - Design UI/UX
  - Find and fix issues

Current Officers

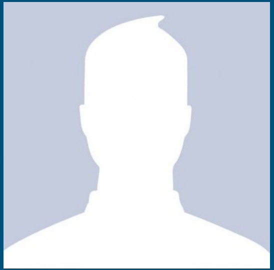
Jermaine

Ryan

Tiger

Keller

Matthew


Alok

Feel free to message us in the #compserv channel on slack, or dm if you prefer to stay private

Candidate Project

Full candidate project info is here: CompServ Candidate Doc

TL;DR:

- Attend Onboarding
- Attend 1 group worksession
- Attend the committee and group socials
- Finish 1 task related to the website (more on this on next slide)

Candidate Project Tasks

- Select your preferences for the following projects:

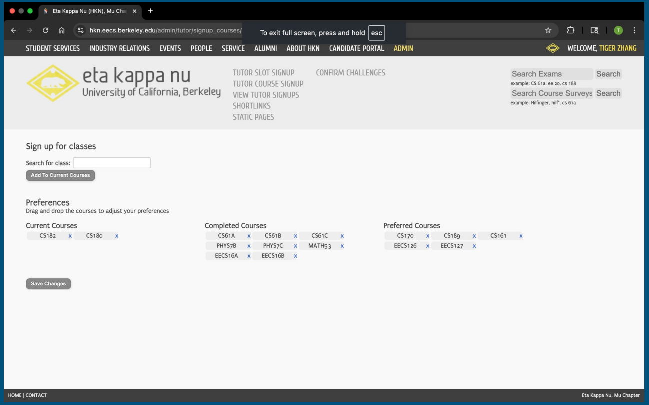
  1. Tutoring Course Pref
  2. Tutoring Sign up
  3. Short Links Page (+ CSV Export and Import)

- If you don’t like any of these feel free to come up with your own (or ask us to make better project ideas i guess)

Tutoring Preferences Page

Description:

- Migrate over the functionality from the old website for officers to choose courses that they would like to tutor for.
- Officers should be able to choose Current Courses, Completed Courses, and Preferred Courses.
- Stay consistent with the new aesthetics of the website.



Sign up for classes

Search for class:

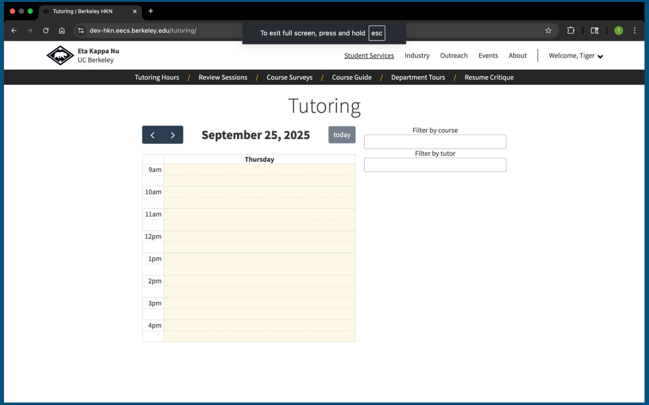
Add To Current Courses

Preferences

Drag and drop the courses to adjust your preferences

Current Courses	Completed Courses	Preferred Courses
CS182 x, CS180 x	CS61A x, CS61B x, CS61C x, PHYS7B x, PHYS7C x, MATH53 x, EECS16A x, EECS16B x	CS170 x, CS189 x, CS161 x, EECS126 x, EECS127 x

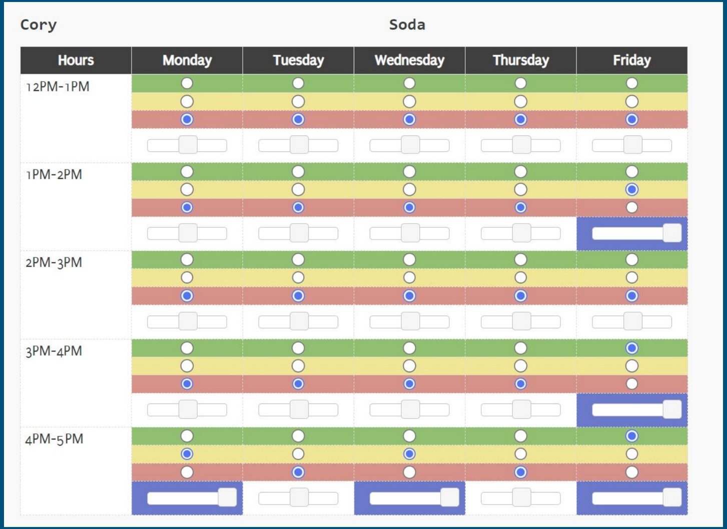
Save Changes



Tutoring Sign Ups Page

Description:

- 2 person project
  - 1 frontend/1 backend?
- Move the old tutoring page to the new site
- Time range in the table should be adjustable



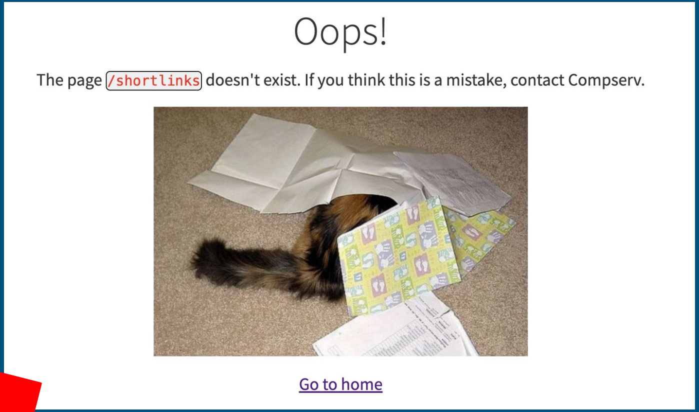
Cory Soda

Hours	Monday	Tuesday	Wednesday	Thursday	Friday
12PM-1PM	○ ○ ●	○ ○ ●	○ ○ ●	○ ○ ●	○ ○ ●
1PM-2PM	○ ○ ●	○ ○ ●	○ ○ ●	○ ○ ●	○ ● ○
2PM-3PM	○ ○ ●	○ ○ ●	○ ○ ●	○ ○ ●	○ ○ ●
3PM-4PM	○ ○ ●	○ ○ ●	○ ○ ●	○ ○ ●	● ○ ○
4PM-5PM	○ ● ○	○ ○ ●	○ ● ○	○ ○ ●	● ○ ○

Short Links Page

Description:

- Bootleg HKN version of bit.ly; ex, hkn.mu/compserv
- Implement the shortlinks page!
  - Allow for creation of new shortlinks
  - List all shortlinks on the page, including metadata of who created them
  - CSV dump for shortlink portability



Oops!

The page /shortlinks doesn't exist. If you think this is a mistake, contact Compserv.

Go to home

this could be a happy cat with your help

hknweb

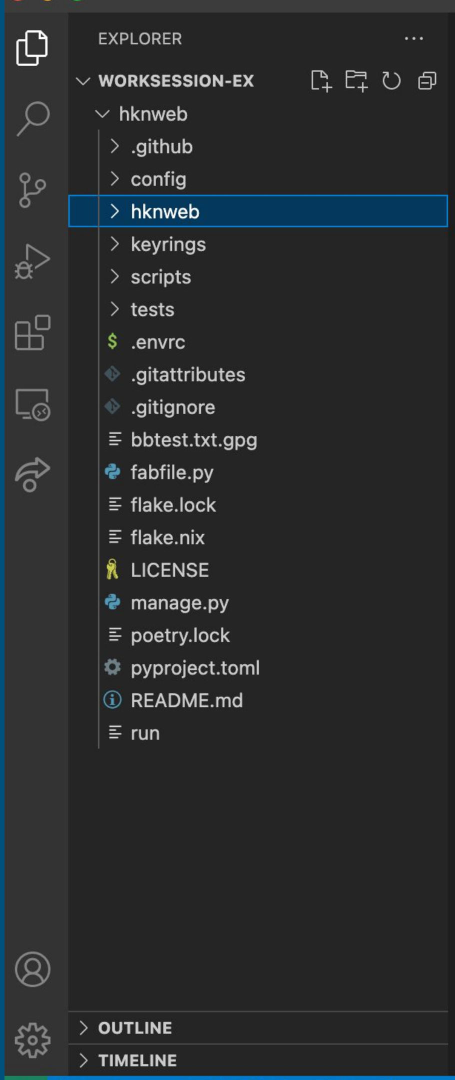
- dev-hkn.eecs.berkeley.edu/
- Our Django-based internal website
- What you’re using for the candidate semester
- Does not contain things like tutoring hours or course surveys - those are on the ruby on rails site at hkn.mu!
- Github repo is here: https://github.com/compserv/hknweb

Setting up hknweb

- Go to the repo at https://github.com/compserv/hknweb
- Fork the repo (button in top left)
- Clone the repo to local (‘git clone <insert http/ssh url here>’ in terminal)
- Install Docker
- cd into the hknweb repo and run ‘docker compose up -d’
- Run ‘docker compose exec app bash’
- Run ‘python manage.py migrate’
- Run ‘python manage.py runserver’ - You should now be able to see the website at http://localhost:8000

hknweb layout

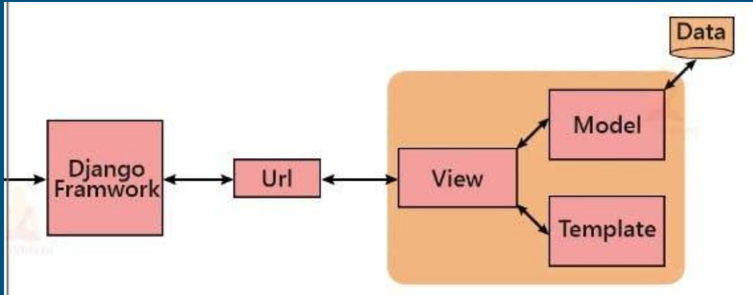
- Most of what you need is within hknweb
- Shouldn't need to touch any of the config/settings files
- Maybe testing if you want to write tests



```
hknweb
  .github
  config
  hknweb
  keyrings
  scripts
  tests
  .envrc
  .gitattributes
  .gitignore
  bbtest.txt.gpg
  fabfile.py
  flake.lock
  flake.nix
  LICENSE
  manage.py
  poetry.lock
  pyproject.toml
  README.md
  run
```

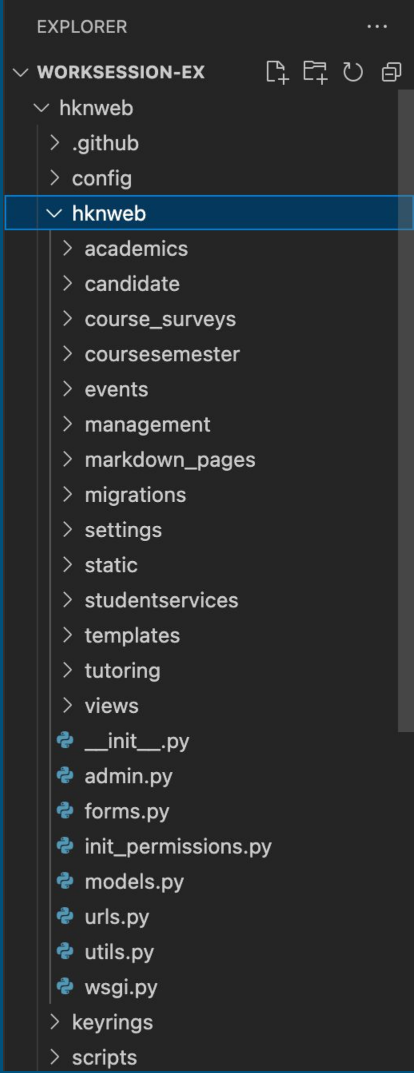
More on Django

- Model-View-Controller framework -> Model, Template, View
- Model - Essentially python classes, defines the structure of the database and relationships between data
- Template - html file that determines the format of the data the user sees
- View - takes in a web request and returns a response. Contains the logic to determine what kind of response should be given
- urls.py - router that maps the url given by the user to a specific view



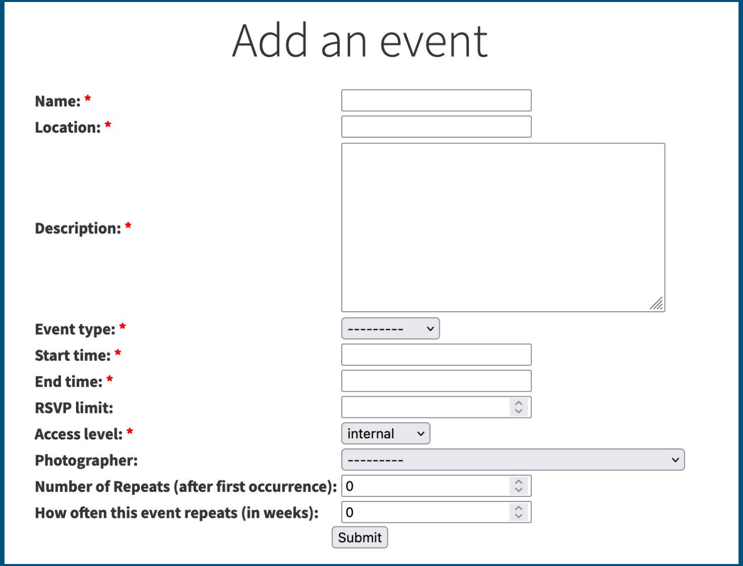
Within hknweb/hknweb

- academics/, candidate/, course_surveys/, coursesemester/, events/, markdownpages/, studentservices/, tutoring/ - different apps each containing their own models and views
- settings/ - config stuff goes here. If you wanted to create a new app, you would edit settings/common.py.
- static/ - static images for display
- templates/ - contains the html layout for each app to render
- views/ - handles the data to pass to each template (there is also a views/ folder within each app for app specific data)
- urls.py - routes the user to pages depending on the url


```
EXPLORER
WORKSESSION-EX
  hknweb
    .github
    config
    hknweb
      academics
      candidate
      course_surveys
      coursesemester
      events
      management
      markdown_pages
      migrations
      settings
      static
      studentservices
      templates
      tutoring
      views
      __init__.py
      admin.py
      forms.py
      init_permissions.py
      models.py
      urls.py
      utils.py
      wsgi.py
    keyrings
    scripts
```

Let's make an update!

- We want to add a "point of contact" text field to an event
- We will need to update the event model, view, and template
- Current layout
- (http://127.0.0.1:8000/events/new?)



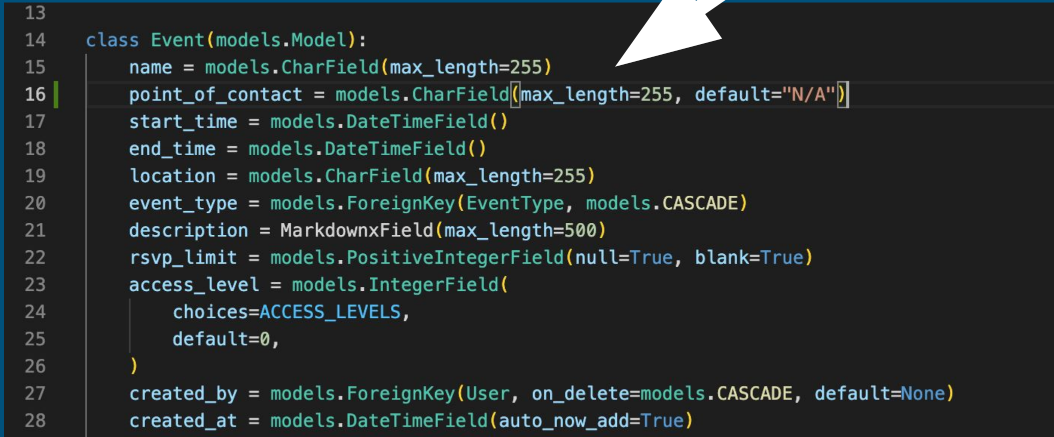
Add an event

Name: *	
Location: *	
Description: *	
Event type: *	---------
Start time: *	
End time: *	
RSVP limit:	
Access level: *	internal
Photographer:	---------
Number of Repeats (after first occurrence):	0
How often this event repeats (in weeks):	0

Submit

Editing events/models/events.py

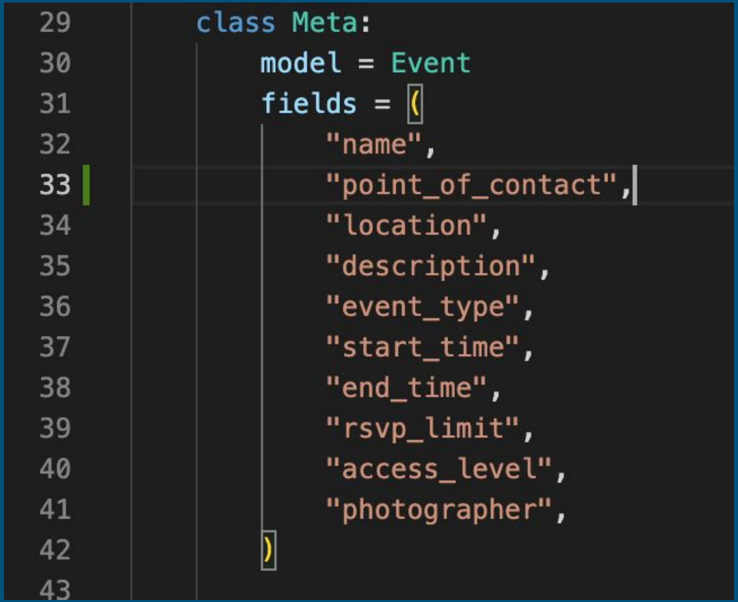
- In line 16, let’s add a new field to the event class called ‘point_of_contact’



```
13
14  class Event(models.Model):
15      name = models.CharField(max_length=255)
16      point_of_contact = models.CharField(max_length=255, default="N/A")
17      start_time = models.DateTimeField()
18      end_time = models.DateTimeField()
19      location = models.CharField(max_length=255)
20      event_type = models.ForeignKey(EventType, models.CASCADE)
21      description = MarkdownxField(max_length=500)
22      rsvp_limit = models.PositiveIntegerField(null=True, blank=True)
23      access_level = models.IntegerField(
24          choices=ACCESS_LEVELS,
25          default=0,
26      )
27      created_by = models.ForeignKey(User, on_delete=models.CASCADE, default=None)
28      created_at = models.DateTimeField(auto_now_add=True)
```

Also edit events/forms/event/create.py

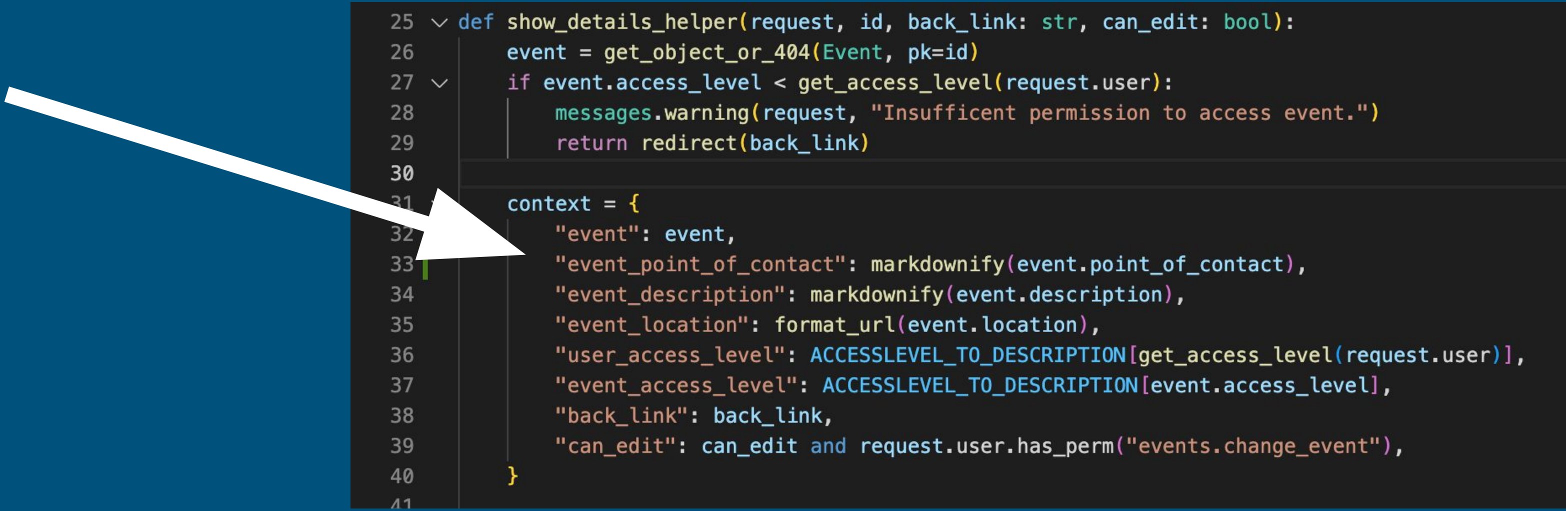
- And events/forms/event/update.py
- Add the “point of contact” field to meta
- Now our database contains the field for the point of contact data
- We will need to let the database know that there has been a change – more on this later!



```
    class Meta:
        model = Event
        fields = (
            "name",
            "point_of_contact",
            "location",
            "description",
            "event_type",
            "start_time",
            "end_time",
            "rsvp_limit",
            "access_level",
            "photographer",
        )
```

Editing events/views/event_transactions/show_event.py

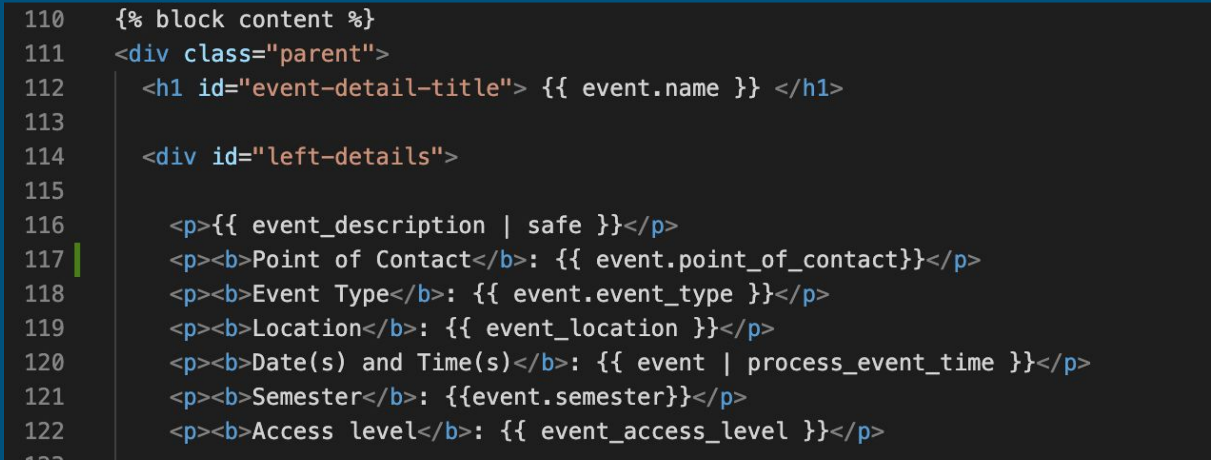
- In line 33, we add an “event_point_of_contact” field
- Now we can pass data from the model to the template



```
25  def show_details_helper(request, id, back_link: str, can_edit: bool):
26      event = get_object_or_404(Event, pk=id)
27      if event.access_level < get_access_level(request.user):
28          messages.warning(request, "Insufficent permission to access event.")
29          return redirect(back_link)
30
31      context = {
32          "event": event,
33          "event_point_of_contact": markdownify(event.point_of_contact),
34          "event_description": markdownify(event.description),
35          "event_location": format_url(event.location),
36          "user_access_level": ACCESSLEVEL_TO_DESCRIPTION[get_access_level(request.user)],
37          "event_access_level": ACCESSLEVEL_TO_DESCRIPTION[event.access_level],
38          "back_link": back_link,
39          "can_edit": can_edit and request.user.has_perm("events.change_event"),
40      }
41
```

Editing templates/events/show_details.html

- How did we know to edit this?
- We add a new line for the point of contact data to be published



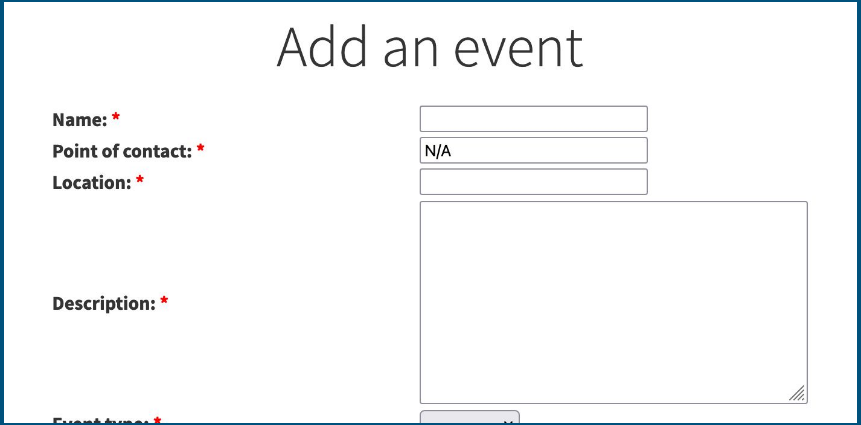
```
{% block content %}
<div class="parent">
  <h1 id="event-detail-title"> {{ event.name }} </h1>

  <div id="left-details">

    <p>{{ event_description | safe }}</p>
    <p><b>Point of Contact</b>: {{ event.point_of_contact}}</p>
    <p><b>Event Type</b>: {{ event.event_type }}</p>
    <p><b>Location</b>: {{ event_location }}</p>
    <p><b>Date(s) and Time(s)</b>: {{ event | process_event_time }}</p>
    <p><b>Semester</b>: {{event.semester}}</p>
    <p><b>Access level</b>: {{ event_access_level }}</p>
```

Final changes

- Since we made an edit to the models, the database tables are now missing the ‘point_of_contact’ field!
- We run ‘python manage.py makemigrations’ and ‘python manage.py migrate’ to update the database with the new field.
- We also need to create an admin account so we can create an event locally, which we can do with ‘python manage.py createsuperuser’
- Now let’s go to the admin portal at http://127.0.0.1:8000/admin and create a new event type - this is required for us to create an event

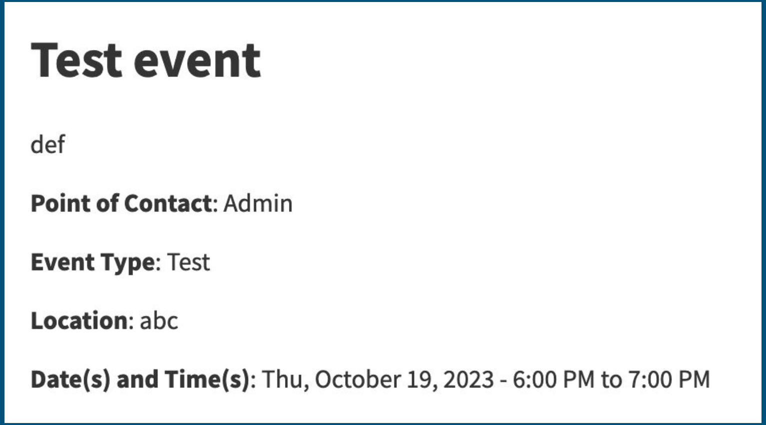
Final result

- We should now be able to create a new event on our local server with the point of contact field
- When we view the event, it also has the point of contact listed in its description



Add an event

Name: *	
Point of contact: *	N/A
Location: *	
Description: *	



Test event

def

Point of Contact: Admin

Event Type: Test

Location: abc

Date(s) and Time(s): Thu, October 19, 2023 - 6:00 PM to 7:00 PM

Submitting changes to hknweb

- Our code is done (and tested), what now?
- Git add/commit/push to your fork (either to master branch or make a new branch)
- Go to https://github.com/compserv/hknweb/pulls and make a new pull request
- Congrats, you have made a pr! This is how you will be submitting your candidate project task (assuming your task is coding related).

Final thoughts

- Let us know if you have any questions! Message us on the candidate slack in the #compserv channel or by dm
- The code for this demo can be found at https://github.com/lo-maxwell/hknweb/tree/worksession-fa23
    - Note that there is an extra file change in the migrations section, this will be automatically generated by running makemigrations/migrate.