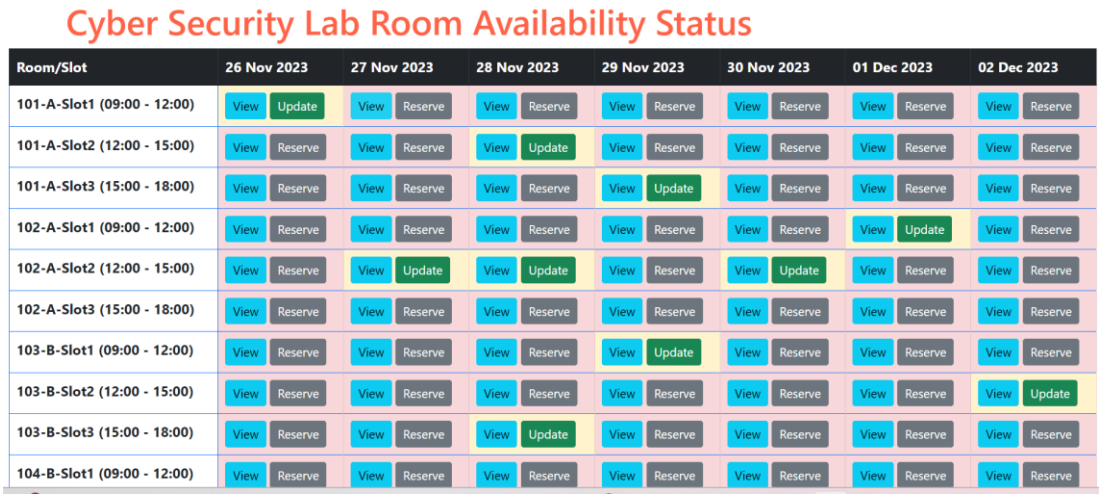
Helpful Instructions for Creating a calendar view Dashboard from a database.

Input database:

This database is a list of students who have reserved slots (slot_id). These reservations are done by the users (user_id). Here our goal is to make a dashboard from this database. The range of slots is from 2 to 16 (15 fixed slots for each data). Hence, any user can book a slot (among 15 slots) for any date. Each Slot and Date is unique. A single slot on a particular date cannot be reserved by any student. The User ID is used when a user logged into the application and reserved a slot. Without user ID no slot can be reserved.

id	name	email	major	purpose	date	slot_id	user_id
30	Student4	stu4@uccs.edu	PhD-CS	Research	2023-11-26	2	5
31	Student4	stu4@uccs.edu	PhD-CS	Research	2023-11-28	3	5
32	Student5	stu5@uccs.edu	MS-CS	Study	2023-11-29	4	5
33	Student5	stu5@uccs.edu	MS-CS	Research	2023-11-27	6	5
34	Student2	stu2@uccs.edu	PhD-Sec	Study	2023-11-25	3	3
35	Student2	stu2@uccs.edu	PhD-Sec	Research	2023-11-28	6	3
36	Student6	stu6@uccs.edu	MS-Sec	Study	2023-11-30	6	3
37	Student7	stu7@uccs.edu	MS-Sec	Research	2023-12-01	5	3
42	Student5	stu5@uccs.edu	PhD-CS	Research	2023-11-29	8	6
43	Student5	stu5@uccs.edu	PhD-CS	Study	2023-11-28	10	6
44	Student7	stu7@uccs.edu	PhD-Sec	Study	2023-12-02	9	6

Output Dashboard:



The Pseudocode for extracting data from database in the views:

- Retrieves all lab rooms from the database (query: labroom)
- Obtains the current date (var: today)
- Create a list of dates for the next seven days starting from today (list: dates)
- Creates a list of lab rooms for iteration (query: labroom > list: labroom_list)
- Create a list of dictionaries (list: urls[{view_url},{update_url}{reserve_url}])
- Generate the lab room detail URL for a specific lab room and date for view url for each slot no matter it is reserved/available. (dict: view_url)
- Generates update and reserve URLs based on whether a student has reserved a slot for the lab room and date. (dict: {update_url} or dict: {reserve_url})
- A loop iterates over each lab room and date to populate the list.
- The table_data list stores dictionaries with lab room information and their corresponding URLs for each date (list: table_data[list:urls])
- The dictionary (dict:context) contains the data to be passed to the template (list:dates, list:labroom_list, list: table_data[list:urls])
- Finally, the template is rendered with the provided context.

The view function is as below:

```
def labroom_list(request):
    labrooms = LabRoom.objects.all()
    today = datetime.now().date()
    dates = []
    labroom_list = [labroom for labroom in labrooms]
    table_data = []

    for i in range(7):
        dates.append(today)
        today = datetime.now().date() + timedelta(days=i+1)
    view_urls_datewise = []

    for labroom in labrooms:
        today = datetime.now().date()
        temp = []
        for i in range(7):
            vur_url = {}
            vur_url["view"] = reverse('labroom-detail', args=[labroom.pk,
str(today)])
            student = Student.objects.filter(slot_id = labroom.id,
date=today).values().all()
```

```
            if student.exists():
                vur_url["update"] = (reverse('update_slot', args=[labroom.pk,
str(today)]))
            else:
                vur_url["reserve"] = (reverse('reserve_slot', args=[labroom.pk,
str(today)]))

            temp.append(vur_url)

            today = datetime.now().date() + timedelta(days=i+1)
        view_urls_datewise.append(temp)
        table_data.append({'labroom':labroom,'urls':temp})

    context = {'views_url':
view_urls_datewise,'dates':dates,'labrooms':labroom_list,'data':table_data}
    return render(request, 'alras_application/labroom_list.html', context)
```

The pseudocode for displaying the dashboard in html:

- Iterate through the dates to generate the table headers.
- Iterate through the row to generate the data rows.
- The first column in the data row prints the room slots.
- The later columns in the data row print the URLs.
- Use a if statement to choose the cell color depending on if the url is for update or reserve.

The html template is as below:

```
{% block content %}
<div class="container justify-content-center">
<h1 style="color:tomato;">Cyber Security Lab Room Availability Status</h1>
</div>
<div class="table-responsive">
{% if data %}
<table class="table table-bordered border-primary">
    <thead class="table-dark">
        <tr>
            <th scope="col">Room/Slot</th>
            {% for date in dates %}
                <th scope="col">{{date|date:"d M Y" }}</th>
            {% endfor %}
        </tr>
    </thead>
```

```
    <tbody>
        {% for x in data %}
        <tr>
            <th class="table-dark-bg-subtle" scope="row">{{x.labroom}}</th>
            {% for url in x.urls %}
                {% if url.reserve %}
                <td class="table-danger"><a class="btn btn-info btn-sm"
href="{{ url.view }}" role="button">View</a>
                   <a class="btn btn-secondary btn-sm" href="{{ url.reserve }}"
role="button">Reserve</a></td>
                {% else %}
                    <td class="table-warning"><a class="btn btn-info btn-sm"
href="{{ url.view }}" role="button">View</a>
                    <a class="btn btn-success btn-sm" href="{{ url.update }}"
role="button">Update</a></td>
                {% endif %}
        {% endfor %}
        </tr>
        {% endfor %}
  </tbody>
</table>
{% else %}
<p>There are no lab rooms configured.</p>
</div>
{% endif %}
</div>
{% endblock %}
```