No Name3
Kanvas with a K
CMPE 131
Assignment 8: CODE REVIEW

Reading Plan:

This code review will describe the functionality of the Kanvas application's task creation; a use case involving a user creating a task will also be described.
Each user is capable of creating a task by navigating to the task creator page from the User dashboard. The purpose of tasks is to remind users individuals or groups of work that needs to be done, and impending deadlines. This is especially useful for group projects where many members need to be managed by a team leader, and when the project is convoluted and needs many tasks to be finished.

Use Case:

- Assuming that: The User has successfully logged in and now resides at the user dashboard page, there should be a list of links to functions in which task creator is one of them. The user clicks the link for task creator.
- A new page is loaded called New Event, this page prompts the user for the information of their needed task: title, description, Start time (year/month/day hour/minute), End time(year/month/day hour/minute).
- After the user has entered the desired specifications of their task, they can create the task by clicking "Create Event", or they can cancel the process by using the "Back" link.
- The User clicks the "Create Event" link and is brought to a new page which lets them review their newly created task info. The page also validates the creation of a task by displaying success to the user. If the user would like to edit the task they can click Edit which brings them to an Edit task page that is the same as the Create event page. If the user clicks back they are brought to their Event page, which lists all current tasks including the newly created one to the user. They have the option to create a new task from this page.

PLAN:
Files of most Importance :
event_controller.rb (describes the functionality of the event/task creation etc.)
events/_form.html.erb (Describes the view of the user and what information the task requires)

User Dashboard HTML: The following HTML shows the three feature buttons on the dashboard, and the page linked to it. The task creator page is linked to the events/new page, which renders an event form.

```
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">

<!-- jQuery library -->
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>

<!-- Latest compiled JavaScript -->
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
<h1>Welcome To Your User Page</h1>

  <a href="events/new" class="btn btn-info" role="button">Task Creator</a>
  <a href="/demo" class="btn btn-info" role="button">Calendar</a>
  <a href="conversations/index" class="btn btn-info" role="button">Conversations</a>
```

Event Form HTML: This page is rendered when the Task creator link is pressed from the User Dashboard. The User has the ability to enter task information on this page.

```
<%= form_for(event) do |f| %>
  <% if event.errors.any? %>
    <div id="error_explanation">
      <h2><%= pluralize(event.errors.count, "error") %> prohibited this event from being
 saved:</h2>

      <ul>
      <% event.errors.full_messages.each do |message| %>
        <li><%= message %></li>
      <% end %>
      </ul>
    </div>
  <% end %>

  <div class="field">
    <%= f.label :title %>
```

```
    <%= f.text_field :title %>
  </div>

  <div class="field">
    <%= f.label :description %>
    <%= f.text_area :description %>
  </div>

  <div class="field">
    <%= f.label :start_time %>
    <%= f.datetime_select :start_time %>
  </div>

  <div class="field">
    <%= f.label :end_time %>
    <%= f.datetime_select :end_time %>
  </div>

  <div class="actions">
    <%= f.submit %>
  </div>
<% end %>
```

Events Controller: This controller class handles the creation, destruction, editing, and indexing of tasks(referred to as events in our code). When a new task is created, the create handler handles saving the user inputted information from the form as a new event in the database. When the User wishes to edit an event, the update controller is used to store the new information. The index action is used to show all of the User's current tasks., displaying all of their specific information such as start and end date.

```
class EventsController < ApplicationController
  before_action :set_event, only: [:show, :edit, :update, :destroy]


  # GET /events
  # GET /events.json
  def index
    @events = Event.all
  end


  # GET /events/1
  # GET /events/1.json
  def show
  end



  # GET /events/new
  def new
    @event = Event.new
  end


  # GET /events/1/edit
  def edit
  end


  # POST /events
  # POST /events.json
```

```ruby
def create
  @event = Event.new(event_params)

  respond_to do |format|
    if @event.save
      format.html { redirect_to @event, notice: 'Event was successfully created.' }
      format.json { render :show, status: :created, location: @event }
    else
      format.html { render :new }
      format.json { render json: @event.errors, status: :unprocessable_entity }
    end
  end
end

# PATCH/PUT /events/1
# PATCH/PUT /events/1.json
def update
  respond_to do |format|
    if @event.update(event_params)
      format.html { redirect_to @event, notice: 'Event was successfully updated.' }
      format.json { render :show, status: :ok, location: @event }
    else
      format.html { render :edit }
      format.json { render json: @event.errors, status: :unprocessable_entity }
    end
  end
end

# DELETE /events/1
# DELETE /events/1.json
```

```ruby
  def destroy
    @event.destroy
    respond_to do |format|
      format.html { redirect_to events_url, notice: 'Event was successfully destroyed.' }
      format.json { head :no_content }
    end
  end

  private
    # Use callbacks to share common setup or constraints between actions.
    def set_event
      @event = Event.find(params[:id])
    end

    # Never trust parameters from the scary internet, only allow the white list through.
    def event_params
      params.require(:event).permit(:title, :description, :start_time, :end_time)
    end
end
```