P2 Backend Implementation

Due Mar 11 2024 by 5 pm EST

https://q.utoronto.ca/courses/337266/assignments/1254644

Requirements

In this phase, you are to implement the Django backend for the project. All your APIs must be RESTful, and you must not have any template or HTML response. You should push your entire Django project to your repo, accompanied by a startup.sh script, a run.sh script, and a docs.pdf file. These files should be located in the root folder of your repository. Note that your code must work on the Ubuntu 20.04.6 LTS. Linux distribution. Consult the Deliverable section for information about how to setup your virtual machine.

- **Startup.sh (Ocean)**: This script should run any preparation needed for your code to run in a new environment. It should create the virtual environment, install all required packages with pip, and run all migrations. More importantly, it should install additional apt packages when necessary, e.g., Python interpreter, the prerequisites for the Python Imaging Library. (pillow), etc.
- **Run.sh (Ocean)**: This script should start your server. In the simplest case, you can just run ./manage.py runserver
- **docs.pdf (Jennifer)**: You must submit a documentation that includes your design of models, as well as the full list of all API endpoints, a short description, their methods, and the payloads. You can use packages that automatically generate the API docs from your code. The TA will send requests based on the information you provide on that document. Therefore, it is important to have a usable and clear document.

OneOnOne/: You should start your project inside this folder.

Ideally, you should not touch your backend server at phase 3, but this will not happen in practice. However, you should implement the backend views for all user stories described in phase 1's handout. Moreover, you should implement proper authentication and authorization for the APIs.

Packages

You are required to use the following packages:

- Django
- Python Imaging Library (pillow)
- Django REST framework
- Simple JWT

You must use token-based authentication for this project.

You may install other packages, but you should document them.

Planning

- Models: (Robert, Jennifer, Kailas)
    - User: use Django built-in model (username, firstname, lastname, email)
    - Event: inviter, invitee, name
        - (consider: how to classify regular/repeating series, and each individual occurrence)
    - **Schedule**: User , Event (foreign Key), availabilities (DateTime object, Foreign key referring back to user)
        - If user has availability linked to it at that time, then the user is free
        - User specifies a free time/availability Datetime object, then it gets added to availabilities
    - **MeetingInvite**: sender (User), recipient (User), meetingLink, meetingName (ForeignKey Event), deadline

Views (ocean):

Accounts App (done)

Views:

- register - For user registration.
- login - For user authentication.
- logout - For logging out users.
- profile - For displaying and editing user profile information.
- change_password - For users to change their password.

Contacts App (half done)

Models:

- UserContact (done)
- MeetingInvite
- Availability

Views:

- ListContactsView (done): Displays a user's list of contacts, allowing them to see who they have added and potentially their availability status for quick scheduling reference.
- AddContactView (done): Enables a user to add a new contact by providing the contact's email address. If the contact is already a registered user, the system would link them; otherwise, it could send an invitation to join the app.
- RemoveContactView (done): Allows users to remove a contact from their list, potentially with confirmation to prevent accidental deletions.
- ContactDetailView (done): Offers detailed information about a specific contact, including their availability if shared, and any upcoming meetings scheduled with them.
- **AddContactAvailability:** Users shoudl be able to update their availability preferences through an interactive week calendar.
- **UpdateContactAvailability**
- UpdateContactPreferencesView: For contacts that are registered users, this view would allow them to update their preferences for meeting times directly in the app, rather than through email interaction.
- ViewContactInvitationsView: Enables users to see pending meeting invitations from other users, allowing them to accept, decline, or suggest alternative times based on their availability.
- SendReminderView: A view for sending reminders to contacts who haven't responded to a meeting invitation, ensuring that users can follow up and finalize schedules efficiently.

Scheduling App

Models:

- Calendar (Done)

- Meeting
- Invitation
- Event

Views:

- calendar_view - Displays the user's calendar with available time slots.
- meeting_create - Form to start the process of scheduling a new meeting, including selecting invitees.
- invitation_response - For invitees to respond to meeting invitations.
- event_finalize - For finalizing a meeting time and converting it into an event.
- event_list - List of upcoming and past events for the user.
- event_detail - Detailed view of a specific event, including time, participants, and any additional information.
- schedule_suggestion - Suggests possible meeting times based on participants' availability.
- reminder_send - To remind invitees who haven't responded to the invitation.

Project Structure

Account/contacts app

1. Upon registration, users can choose their availabilities which will apply to all events and all calendars
2. Each availability has a rank based on preference
3. Each user has a list of contacts

Scheduling app:

4. Each user can create an event and meeting calendar, and send an invite to users to join this meeting calendar
5. Each event has a meeting calendar, each calendar contains meetings, each meeting contains 1 organizer and 1 participant
6. Each event/meeting calendar has 1 organizer and list of participants
    a. Each event/meeting calendar has a start/end date
7. Each meeting has start/end time, and date
8. Organizers can generate a list of suggested meeting calendars
    a. Suggested meeting calendar will contain the maximal matching

        i. Bipartite matching between A = {organizer} and B = {event participants} *(maybe we can use bipartite matching max flow?)*
    b. If there is no suggested meeting calendar that respects all availability constraints, then just return maximal matchings, and users that cannot be matched in a separate list to the side
9. Organizers can finalize meetings based on generated suggested calendars
10. Once calendar finalized, create meetings with unique <meeting_id> and corresponding start/end times, and send meeting link to participant and inviter

Advice

- Start this phase by modeling your application. A **simple class diagram or ER diagram** will go a long way for your project. Please consult the Assignment 2 Handout for some ideas on how to describe your endpoints.
- The basic User model provided by Django will likely be sufficient for your needs. If you use a custom User model, make sure you derive from the AbstractUser base class to inherit Django's authentication system. You are suggested look into Customizing Django's User Model for more detail.

Deliverable

Outside of the grading session, TAs will clone and run your codes on a virtual machine with Ubuntu 20.04.6 LTS. You may want to consult this guide on how to setup an Ubuntu desktop on VirtualBox. This is the environment in which your code will be run in phases 2 and 3, so you are advised to setup the virtual machine to run your code. You are allowed to work on other operating systems and environments as well, but at your own risk. It is crucially important for you to at least double-check that your whole project works in the virtual machine before submission. You will not be allowed to modify your code at the interview sessions. Therefore, if, for any reason, your code does not run, you will not get any marks.

Marking

You should use the CSC309 Booking System to reserve your interview session with a TA. Please note that if you do not book a grading session, you will get no marks for that phase. All team members must be present at the meeting, and the absent members will not get any marks for the phase. You must complete your interview no later than 10 days after the submission deadline.

Each team member must do a fair amount of work on this project. We will assess each member's participation by investigating the commits they created and asking detailed questions about the code at the interviews. In most cases, all team members will receive the same mark. However, we reserve the right to give individual marks in cases where we observe unfair distribution of work.

The TA will ask you to use Postman to demonstrate each of your API endpoints on your machine. Before your grading session, it is important that you prepare a comprehensive set of test data that is capable of demonstrating various aspects of your backend, e.g. search filter, different reservation states, ability to paginate. You will need enough data set to show that each component is working. You will lose marks if you run out of time during your grading session, so please come prepared.