Docs

Accounts

Models

- **UserSetting**
  - Fields:
    - `id` (UUIDField): A unique identifier for each user setting, generated using UUID4.
      - `user` (OneToOneField): Links to the Django's built-in User model. It ensures a one-to-one relationship between the user and their settings.

Serializers

- **UserSerializer**
  - Fields: `username`, `email`, `first_name`, `last_name`, `password`.
  - Methods: `create` to register a new user, `validate_password` to ensure password meets certain criteria.
- **EditProfileSerializer** (Inherits from UserSerializer)
  - Fields: Same as UserSerializer.
  - Overrides: `validate_email` to check email validity, `update` to allow editing user profile.

API Endpoints

1. **Signup**
   - URL: `/signup/`
   - Method: `POST`
   - Description: Allows new users to create an account.
   - Payload: `{ "username": "", "email": "", "first_name": "", "last_name": "", "password": "" }`
2. **Login**
   - URL: `/login/`
   - Method: `POST`
   - Description: Authenticates a user.
   - Payload: `{ "username": "", "password": "" }`
3. **Profile**
   - URL: `/profile/`
   - Method: `GET`
   - Description: Retrieves the profile information of the currently authenticated user.
   - Authentication: Required
4. **Edit Profile**
   - URL: `/profile/edit/`
   - Method: `POST`

- Description: Allows users to edit their profile information.
- Payload: `{ "username": "", "email": "", "first_name": "", "last_name": "", "password": "" }`
- Authentication: Required

5. **Delete Account**
    - URL: `/profile/delete/`
    - Method: `POST`
    - Description: Allows users to delete their account.
    - Payload: `{ "refresh_token": "" }`
    - Authentication: Required
6. **Logout**
    - URL: `/logout/`
    - Method: `POST`
    - Description: Logs out the user by blacklisting the refresh token.
    - Payload: `{ "refresh_token": "" }`
    - Authentication: Required
7. **Token Obtain Pair**
    - URL: `/token/`
    - Method: `POST`
    - Description: Obtains a new JWT token pair.
    - Payload: `{ "username": "", "password": "" }`
8. **Token Refresh**
    - URL: `/token/refresh/`
    - Method: `POST`
    - Description: Refreshes an existing JWT token.
    - Payload: `{ "refresh": "" }`
9. **Reset Password Request**
    - URL: `/password_reset/`
    - Method: `POST`
    - Description: Initiates a password reset request.
    - Payload: `{ "email": "" }`
10. **Reset Password**
    - URL: `/password_reset/<uidb64>/<token>/`
    - Method: `POST`
    - Description: Completes the password reset process.
    - Payload: `{ "password": "" }`

G

Calendars

Models

- **Calendar**
    - `id`: Unique identifier using UUID.
    - `owner`: Foreign Key linking to the User model, representing the calendar's owner.
    - `name`: The name of the calendar.
    - `is_finalized`: Boolean indicating if the calendar's schedule is finalized.
- **CalendarParticipant**
    - `id`: Unique identifier using UUID.
    - `calendar`: Foreign Key linking to the Calendar model.
    - `user`: Foreign Key linking to the User model, representing a participant of the calendar.
- **NonBusyTime**
    - `id`: Unique identifier using UUID.
    - `user`: Foreign Key linking to the User model.
    - `calendar`: Foreign Key linking to the Calendar model.
    - `start_time` and `end_time`: DateTime fields defining the start and end of a non-busy time slot.
    - `preference_level`: An integer indicating the preference level of the non-busy time slot.
- **Meeting**
    - `id`: Unique identifier using UUID.
    - `calendar`: Foreign Key linking to the Calendar model.
    - `final_time`: DateTime field for the finalized meeting time.
    - `deadline`: DateTime field for the deadline to finalize the meeting time.
- **ScheduleSuggestion**
    - `id`: Unique identifier using UUID.
    - `meeting`: Foreign Key linking to the Meeting model.
    - `suggested_time`: DateTime field for the suggested time of the meeting.
- **Invitation**
    - `id`: Unique identifier using UUID.
    - `calendar`: Foreign Key linking to the Calendar model.
    - `sender` and `receiver`: Foreign Keys linking to the User model.
    - `status`: Status of the invitation (pending, accepted, declined).

API Endpoints

Base URL Prefix: `/calendars/`

Make sure you are authenticated by using a bearing token in your requests

1. Calendar Management

- **Create a Calendar**
  - **Method**: `POST`
  - **Endpoint**: `/calendars/`
  - **Payload**:

    ```
    {
      "name": "{calendar_name}",
      "is_finalized": {is_finalized_boolean}
    }
    ```

    - Replace `{calendar_name}` with the desired name of the calendar.
    - Replace `{is_finalized_boolean}` with `true` or `false` to indicate if the calendar is finalized.
- **Update a Calendar**
  - **Method**: `PUT`
  - **Endpoint**: `/calendars/{calendar_id}/`
  - **Payload**:

    ```
    {
      "name": "{updated_calendar_name}",
      "is_finalized": {updated_is_finalized_boolean}
    }
    ```

    - Replace `{updated_calendar_name}` with the new name for the calendar.
    - Replace `{updated_is_finalized_boolean}` with `true` or `false` to update the finalized status.

2. Calendar Participants Management

- **Add a Participant**
  - **Method**: `POST`
  - **Endpoint**: `/calendars/{calendar_id}/participants/`
  - **Payload**:

    ```
    {
      "user": {user_id}
    }
    ```

    - Replace `{user_id}` with the ID of the user to be added as a participant.

3. Non-Busy Times Management

- **Add Non-Busy Time**
  - **Method**: `POST`

- **Endpoint**: `/calendars/{calendar_id}/nonbusytimes/`
- **Payload**:

```
jsonCopy code
{
  "start_time": "{start_time_iso}",
  "end_time": "{end_time_iso}",
  "preference_level": {preference_level_int}
}
```

  - Replace `{start_time_iso}` and `{end_time_iso}` with the start and end times in ISO 8601 format.
  - Replace `{preference_level_int}` with an integer indicating the preference level.

4. Meetings Management

- **Schedule a New Meeting**
  - **Method**: `POST`
  - **Endpoint**: `/calendars/{calendar_id}/meetings/`
  - **Payload**:

```

{
  "deadline": "{deadline_iso}"
}
```

    - Replace `{deadline_iso}` with the deadline for finalizing the meeting in ISO 8601 format.
- **Update a Meeting**
  - **Method**: `PUT`
  - **Endpoint**: `/calendars/{calendar_id}/meetings/{meeting_id}/`
  - **Payload**:

```

{
  "final_time": "{final_time_iso}",
  "deadline": "{updated_deadline_iso}"
}
```

    - Replace `{final_time_iso}` with the new finalized time for the meeting in ISO 8601 format.
    - Replace `{updated_deadline_iso}` with the updated deadline in ISO 8601 format

- **Suggest a New Time for a Meeting**
  - **Method**: `POST`
  - **Endpoint**: `/calendars/meetings/{meeting_id}/suggestions/`
  - **Payload**:

```
{
  "suggested_time": "{suggested_time_iso}"
}
```

    - Replace `{suggested_time_iso}` with the suggested time for the meeting in ISO 8601 format.

- **Delete a Schedule Suggestion**
  - **Method**: `DELETE`
  - **Endpoint**: `/calendars/meetings/{meeting_id}/suggestions/{suggestion_id}/`
  - **Description**: Removes a specific schedule suggestion.

6. Invitations Management

- **Send an Invitation to Join a Calendar**
  - **Method**: `POST`
  - **Endpoint**: `/calendars/invitations/`
  - **Payload**:

```
{
  "calendar": "{calendar_id}",
  "receiver": {receiver_user_id},
}
```

    - Replace `{calendar_id}` with the ID of the calendar to which you're inviting the user.
    - Replace `{receiver_user_id}` with the ID of the user being invited. You can accomplish this by making a `GET` request to `/calendars/get-user-id/<username>`
    - Replace `{invitation_status}` with the initial status of the invitation, typically `"pending"`.

- **Delete an Invitation**
  - **Method**: `DELETE`
  - **Endpoint**: `/calendars/invitations/{invitation_id}/`
  - **Description**: Revokes or deletes an invitation.

7. Send Email Notification

- **Send Email**
  - **Method**: `POST`
  - **Endpoint**: `/calendars/send-email/`
  - **Payload**:

```
{
  "username": "{username}",
}
```

        - Replace `{username}` with the username of the user to whom the email will be sent.

Contacts

Base URL Prefix: `/contacts/`

1. Contact List Management

- **List Contacts**
    - **Method**: `GET`
    - **Endpoint**: `/contacts/`
    - **Description**: Retrieves a list of all contacts associated with the authenticated user.
    - **Payload**: No payload required for GET requests.

2. Add a New Contact

- **Add Contact**
    - **Method**: `POST`
    - **Endpoint**: `/contacts/add/`
    - **Payload**:

      ```
      {
        "contactee": "{contactee_username}"
      }
      ```

        - Replace `{contactee_username}` with the username of the user you want to add as a contact.
    - **Description**: Adds a new contact for the authenticated user, using the contactee's username to identify them.

3. Delete a Contact

- **Delete Contact**
    - **Method**: `DELETE`
    - **Endpoint**: `/contacts/delete/{username}/`
        - Replace `{username}` in the endpoint URL with the username of the contactee you wish to delete.
    - **Payload**: No payload required for DELETE requests.
    - **Description**: Deletes a specific contact from the authenticated user's contacts list, identified by the contactee's username.

Payload Parameters Description:

- For the **Add Contact** endpoint, the payload requires:
    - `contactee`: The username of the user to be added as a contact. This should be the username of an existing user in the system.

Usage Notes:

- All endpoints require authentication. Users must be logged in to interact with these APIs.

- The **List Contacts** endpoint returns all contacts where the authenticated user is the owner.
- The **Add Contact** endpoint requires a valid username of another user in the system to establish a contact relationship.
- The **Delete Contact** endpoint allows for the removal of a contact relationship based on the contactee's username.

These API endpoints provide a concise and structured way to manage contacts within your application, facilitating easy listing, addition, and deletion of contact relationships through your Contacts app's API.