CSC309 Project P2 Documentation

Helen (Jin Yi) Li, Sinan Li, Nicholas Poon, Raghav Srinivasan

November 15, 2023

1 Accounts

`Account Model`

- username: email field that must be unique
- password: string
- first_name: string, at most 50 characters
- last_name: string, at most 50 characters
- is_seeker: boolean, represents whether the user is a seeker (true) or a shelter (false)
- notif_preference: boolean, represents whether the user wants notifications or not (not required, default=true)
- avatar: image field for profile picture with a default image

`POST /api/token/`

- Does NOT require authentication
- Request body fields: username (email), password (string)
- Generates access and refresh tokens to log the user in if the data passed in the body refers to a registered account

`GET /accounts/<int:pk>/`

- Requires authentication
- No request body needed
- Seekers are authorized to view their own account and all shelter accounts
- Shelters are authorized to view all shelter accounts and seeker accounts who have an active application with them (i.e., 'pending' status)

`GET /accounts/`

- Requires authentication
- No request body needed
- Retrieves a list of shelters (no endpoint for retrieving a list of seekers, as per the requirements)

`POST /accounts/`

- Does NOT require authentication
- Request body fields: username (email), password (string), reenter_password (string), first_name (string), last_name (string), is_seeker (boolean), notif_preference (boolean), avatar (file)
- Creates a new user account based on the data passed in the body

`PUT /accounts/`

- Requires authentication

- Request body fields: username (email), password (string), first_name (string), last_name (string), is_seeker (boolean), notif_preference (boolean), avatar (file)
- Updates the account of the authenticated user sending the request
- Update an existing account based on request body data (any missing fields means the existing value will be retained)
- Users cannot switch between seeker and shelter account types, so is_seeker will not be updated (it is acceptable to include this field in the request body as long as the new value matches the existing value of the corresponding account)

`DELETE /accounts/`

- Requires authentication
- No request body needed
- Deletes the account of the authenticated user sending the request

2 Pet Listings

`Owner Model`

- name: string, at most 50 characters
- email: email field that must be unique
- phone: string, at most 50 characters
- location: string, at most 50 characters
- birthday: date field

`Pet Listing Model`

- owner: foreign key to owner (cascade on delete)
- shelter: foreign key to shelter (user model) (cascade on delete)
- status: string, at most 50 characters; can be one of the following values: “available”, “adopted”, “pending”, “withdrawn”
- last_update: datetime (use auto_now to track time of most recent change)
- creation_date: date (use auto_now_add to track time of creation)

`Pet Model`

- name: string, at most 50 characters
- gender: string, at most 50 characters; can be one of the following values: “male”, “female”
- birthday: date
- weight: integer, greater than or equal to 0, represents the weight of the pet in kg
- animal: string, at most 50 characters
- breed: string, at most 50 characters
- colour: string, at most 50 characters
- vaccinated: boolean
- other_info: string, at most 50 characters
- pet_listing: foreign key to pet listing (cascade on delete)

`Picture Model`

- pet: foreign key to pet (cascade on delete)
- path: image field that stores a path to the file name in static/pet_listing_pics

- creation_time: datetime (use auto_now_add to track time of creation)

`GET /pet_listings/`

- Requires authentication
- Can only be accessed by a pet shelter account
- Displays the currently logged-in pet shelter's pet listings

`POST /pet_listings/`

- Requires authentication
- Can only be accessed by a pet shelter account
- Allows the currently logged-in pet shelter to create a new pet listing
- Supports a maximum of 5 uploaded pictures (any picture format) per pet listing
- Request body:

```
{
    "pet_name": "[PET_NAME]",
    "gender": ["male", "female"],
    "pet_birthday": "YYYY-MM-DD",
    "pet_weight": "[PET_WEIGHT_(KG)]",
    "animal": "[ANIMAL]",
    "breed": "[BREED]",
    "colour": "[COLOUR]",
    "vaccinated": [true, false],
    "pictures": [*.jpg, *.png, *.jpeg, ...], // at most 5
    "owner_name": "[OWNER_NAME]",
    "email": "[EMAIL]",
    "phone_number": "[PHONE NUMBER]",
    "location": "[LOCATION]",
    "owner_birthday": "YYYY-MM-DD",
    "status": ["available", "adopted", "pending", "withdrawn"]
}
```

`GET /pet_listings/<int:pet_listing_id>/`

- Requires authentication
- Can only be accessed by the pet shelter account that created the pet listing with id `pet_listing_id`
- Allows the currently logged-in pet shelter to view the details of the pet listing with id `pet_listing_id`

`PUT /pet_listings/<int:pet_listing_id>/`

- Requires authentication
- Can only be accessed by the pet shelter account that created the pet listing with id `pet_listing_id`
- Allows the currently logged-in pet shelter to edit the pet listing with id `pet_listing_id`
- Supports a maximum of 5 uploaded pictures (any picture format) per pet listing
- Request body:

```
{
    "pet_name": "[PET_NAME]",
    "gender": ["male", "female"],
    "pet_birthday": "YYYY-MM-DD",
    "pet_weight": "[PET_WEIGHT_(KG)]",
    "animal": "[ANIMAL]",
    "breed": "[BREED]",
    "colour": "[COLOUR]",
    "vaccinated": [true, false],
    "pictures": [*.jpg, *.png, *.jpeg, ...], // at most 5
    "owner_name": "[OWNER_NAME]",
    "email": "[EMAIL]",
    "phone_number": "[PHONE NUMBER]",
    "location": "[LOCATION]",
    "owner_birthday": "YYYY-MM-DD",
    "status": ["available", "adopted", "pending", "withdrawn"]
}
```

  - If the combined total of the current pictures and newly uploaded pictures exceeds 5, the least recently uploaded pictures will be replaced first

`DELETE /pet_listings/<int:pet_listing_id>/`

- Requires authentication
- Can only be accessed by the pet shelter account that created the pet listing with id `pet_listing_id`
- Allows the currently logged-in pet shelter to delete the pet listing with id `pet_listing_id`
- Deletes the corresponding pet, application, and comment upon removal of pet listing

`POST /pet_listings/search_results/`

- Supports 6 filters:
  - **Shelter** (key: shelter)
    - Accepts a list of integers as input where the integers are the ids of valid shelters
  - **Status** (key: status)
    - Accepts a list of strings as input where the strings are 1 or more of the following
      - available
      - pending
      - adopted
      - withdrawn
    - If not provided, only pet listings with an available status will be displayed
  - Gender (key: gender)
    - Accepts a list of strings as input where the strings are 1 or more of the following
      - male
      - female
  - Start Date (key: start_date)
    - Accepts a date field in the format of YYYY-MM-DD
  - End Date (key: end_date)
    - Accepts a date field in the format of YYYY-MM-DD
  - Pet Type (key: pet_type)
    - Accepts a list of strings as input where the strings are 1 or more of the following:
      - dog
      - cat
      - bird
      - other
- Supports 2 sort methods (key: sort)
  - Pet Name

    * The default sorting method
  - Size (Weight)

- Supports pagination (4 results per page)
- Request body:

```
{
    "shelter": Optional[List[int]], // list of shelter ids
    "status": Optional[List["available", "pending", "adopted", "withdrawn"]]; default = ["available"],
    "gender": Optional[List["male", "female"]]
    "start_date": Optional["YYYY-MM-DD"]
    "end_date": Optional["YYYY-MM-DD"]
    "pet_type": List["dog", "cat", "bird", "other"],
    "sort": Optional["weight", "name"]; default = "name"
}
```

`GET /pet_listings/search_results/<int:pet_listing_id>/`

- Allows users to view the details of the pet listing with id `pet_listing_id`

3 Comments

`ShelterComment Model`

- shelter: foreign key to User (cascade on delete) with related name `shelter_comments`
- commenter: foreign key to User (set null on delete)
- comment: message to post to shelter comment section, at most 200 characters
- parent: foreign key to ShelterComment (cascade on delete) with related name `replies`
- date: datetime (use auto now add to track time of creation)

`ApplicationComment Model`

- application: foreign key to User (cascade on delete) with related name `application_comments`
- commenter: foreign key to User (set null on delete)
- comment: message to post to application comment section, at most 200 characters
- parent: foreign key to ApplicationComment (cascade on delete) with related name `replies`
- date: datetime (use auto now add to track time of creation)

`GET /comments/shelter/<int:shelter_id>/`

- Any logged in seeker/shelter can access
- Retrieves a list of comments posted for the given shelter
- Replies to a comment are listed under `replies` field
- Sorted by creation time in descending order (newest at the top)
- Paginated such that each page displays 10 comment threads

`POST /comments/shelter/<int:shelter_id>/`

- Any logged in seeker/shelter can access
- Request body fields: comment (string: content of message to post)
- Creates a new comment based on data passed in the body for the given shelter

`POST /comments/shelter/<int:shelter_id>/<int:comment_id>/`

- Any logged in seeker/shelter can access
- Request body fields: comment (string: content of message to post)
- Creates a new reply based on data passed in the body for the given shelter, that replies to comment given by comment_id
- If comment_id is not a top-level comment, create a new reply to the top-level comment of the thread

`GET /comments/app/<int:app_id>/`

- Only the seeker and shelter of the given app_id can access
- Retrieves a list of comments posted for the given application
- Replies to a comment are listed under `replies` field
- Sorted by creation time in descending order (newest at the top)
- Paginated such that each page displays 10 comment threads

`POST /comments/app/<int:app_id>/`

- Only the seeker and shelter of the given app_id can access
- Request body fields: comment (string: content of message to post)
- Creates a new comment based on data passed in the body for the given application
- Updates last_modified time of given application

`POST /comments/app/<int:app_id>/<int:comment_id>/`

- Only the seeker and shelter of the given app_id can access
- Request body fields: comment (string: content of message to post)
- Creates a new reply from data passed in the body for given application that replies to comment `comment_id`
- If comment_id is not a top-level comment, create a new reply to the top-level comment of the thread
- Updates last_modified time of given application

4 Applications

`Application Model`

- seeker: foreign key to user (cascade on delete)
- shelter: foreign key to user (cascade on delete)
- pet_listing: foreign key to pet listing (cascade on delete)
- status: string, default 'pending', possible application status strings are 'accepted', 'withdrawn', 'pending', or 'denied'
- creation_date: datetime (use auto_now_add to track time of creation)
- last_modified: datetime (use auto_now to track time of most recent change)

`GET /applications/<int:pk>/`

- Requires authentication
- No request body needed
- Retrieves the application with the specified id (seekers can only view their submitted applications while shelters can only view their received applications)

`GET /applications/`

- Requires authentication
- Request body fields: filters (list of application status strings), sort (string: creation_date or last_modified)
- Retrieves a list of applications (seekers can only view their submitted applications while shelters can only view their received applications)
- Supports filtering applications by status
- Supports sorting application by creation time or last update time
- Supports pagination (3 applications per page)

`POST /applications/`

- Requires authentication
- Request body fields: pet_listing (int: id of a pet listing object), status ('accepted', 'withdrawn', 'pending', or 'denied')
- Do NOT need seeker and shelter fields (inferred as the authenticated user and pet listing's shelter, respectively)
- Applications can only be created for available pet listings (i.e., 'available' status)
- Each seeker can only file one application for a pet listing

`PUT /applications/`

- Requires authentication
- Request body fields: id (int: id of an existing application), status ('accepted', 'withdrawn', 'pending', or 'denied')
- The application to be updated must be created (seeker) or received (shelter) by the authenticated user
- Status is the only field that can be updated, so additional fields in the request body are ignored
- Shelters can only update the status of an application from pending to accepted or denied
- Seekers can only update the status of an application from pending or accepted to withdrawn

5 Notifications

5.1 Model

`Notification Model`

- `receiver` - ForeignKey - `Accounts.User`
  The user receiving the notification.
- `message` - ForeignKey - `models.TextField`
  The message of the notification.
- `c_time` - `models.DateTimeField`
  The date and time the notification was created.
- `is_read` - `models.BooleanField`
  Whether the notification has been read.
- `related_link` - `models.URLField`
  The link to the associated model

5.2 End Points

`GET /notifications/`

- List all the notifications related to the currently logged in user
- In order to filter by `is_read`, append `?is_read=True` or `?is_read=False`.

`POST /notifications/`

- Create a notification
- Request body:

```
{
    "recriver": RECEIVER_ID
    "message": "[MESSAGE_BODY]",
    "related_link": "[RELATED_LINK]"
}
```

`GET /notifications/<int:pk>/`

- Show the details of the notification whose `id` is `pk`
- Return 404 if the notification DNE or does not belong to the user

`PUT /notifications/<int:pk>/`

- Update the `is_read` field of the notification whose `id` is `pk` to `True`

`DELETE /notifications/<int:pk>/`

- Deletes a notification whose `id` is `pk`
- Returns 404 if the notification DNE or does not belong to the user