P2 Model and Endpoint Breakdown

The superuser:

Username = superusername

Password = superpassword

Models

User

- The same as the standard User, but it also has a phone number with a phone regex
- The user also has an about field where the user can describe themselves
- The user also has generic relations with comments and notifications where you can access the notifications and comments the user has
- The user has a rating sum and a rating number. rating_sum/rating_number = average rating
- Finally the user has a profile picture field which has a default image at media/profile/default.jpg

Home

- name: the name of the property
- description: description of the property
- baths, beds: the number of these amenities, default 1
- guests: the max number of guests allowed, default 2
- street_address, city, state, country: address of the property
- date_added: automatically set when added
- image: the profile image for the property, default at media/homes/default.png
- owner: a User object
- comments: a reverse relation to see all comments for this property
- rating_sum: the sum of all ratings

- rating_number: the number of ratings

<u>Price</u>
- home: the Home this price range is for
- price: the daily price of the home during the availability
- start_date: the start date of the availability
- end_date: the end date of the availability

<u>Images</u>
- home: the Home this price range is for
- image: the image being added
- alt: description of the image

<u>Reservation</u>
- home: the Home being reserved
- renter: the User reserving the home
- start_date: the start date of the reservation
- end_date: the end date of the reservation (must ensure it is greater than start_date)
- expiry_date: default is the minimum between a week from the day of the reservation, or the start date
- status: the status of the reservation
    - 0: pending
    - 1: denied
    - 2: expired
    - 3: approved
    - 4: canceled
    - 5: pcancel (pending cancelation)
    - 6: terminated
    - 7: completed

<u>Notification</u>

- message: the notification message
- content_type, object_id: details about the object being notified about
- notification_object: the object being notified about
    - When setting the fields, simply set the notification object and content_type and object_id will be updated
- notifee: the User being notified
- notif_link: the url link to the notification (please use reverse_lazy
- read: False if unread, True if read

<u>Comment</u>

- rating: an integer from 0 to 5
- message: the comment
- content_type, object_id: details about the object being commented about
- comment_object: the object being commented about
    - When setting the fields, simply set the comment object and content_type and object_id will be updated
- comentee: the User who made the comment

<u>Response</u>

- message: the response
- content_type, object_id: details about the object being notified about
- comment_object: the object being notified about
    - When setting the fields, simply set the comment object and content_type and object_id will be updated
- comentee: the User who made the reply
- comment: the original comment
- level: the level of the response

ENDPOINTS

Account Endpoints

Endpoint: /accounts/token/
Methods: POST
Payload: username, password

Permissions:
- None

Return: 200 OK with a refresh and access JWT

Endpoint: /accounts/token/refresh/
Methods: POST
Payload: refresh

Permissions:
- None

Return: 200 OK with a new access JWT

Endpoint: /accounts/signup/
Methods: POST
Payload: username, password, about, first_name, last_name, email, phone
Required: username, password, first_name, last_name

Permissions:
- None

Return: 201 Created and the newly created user

Endpoint: /accounts/profile/details/
Methods: GET
Fields: id, first_name, last_name, username, email, phone, about, profile_pic,

Permissions:
- IsAuthenticated: Must pass JWT access token in header

Return: 200 OK and the logged in user

Endpoint: /accounts/profile/edit/
Methods: PUT, PATCH
Payload: first_name, last_name, email, phone, about, profile_pic, password

Permissions:

- IsAuthenticated: Must pass JWT access token in header

Return: 200 OK and the updated logged in user

Endpoint: /accounts/user/<slug:pk>/details/
Methods: GET
Fields: id, first_name, last_name, username, email, phone, about, profile_pic,

Permissions:

- None

Return: 200 OK and the user with id=pk

Endpoint: /accounts/user/<slug:pk>/add_comment/
Methods: POST
Payload: rating, message

Permissions:

- IsAuthenticated: Must pass JWT access token in header
- HasHostedUserPermission: Must have been the owner of a home reserved by the user with id=pk
- NoPrevUserComPermission: Must have *not* made a previous comment about this user with id=pk

Return: 201 Created and the newly created comment
Additional Notes:

- The user being commented about and the user making the comment are automatically passed
- The rating is added to the user's rating_sum and rating_number

Endpoint: /accounts/reservations/
Methods: GET
Fields: id, home, renter, start_date, end_date, expiry_date, status
Payload: status

Permissions:

- IsAuthenticated: Must pass JWT access token in header

Return: 200 OK and the set of the logged in users reservations
Additional Notes:
- If the user passes a value for status in the GET request, the results will be filtered by this status
- Before returning the list, it makes sure there are no expired or completed reservations with statuses that do not match
- There is pagination support. Each page will display 10 results

<u>**Home Endpoints**</u>

Endpoint: /homes/add/
Methods: POST
Payload: id, name, description, baths, beds, guests, street_address, city, state, country, image

Permissions:
- IsAuthenticated: Must pass JWT access token in header

Return: 201 Created and the newly created home
Additional Notes:
- The user is automatically passed as the owner of the home

Endpoint: /homes/search/
Methods: GET
Fields: id, name, description, baths, beds, guests, street_address, city, state, country, image, owner, rating_sum, rating_number
Payload: city, state, country, baths, beds, rating_number, guests, sort_rating_number, sort_baths, sort_beds

Permissions:
- None

Return: 200 OK and a list of homes matching the filters from the payload
Additional Notes:
- The results can be sorted by the following fields from the payload
    - city, state, country, number of baths, number of beds, number of ratings (rating_number), number of guests

- The results can be ordered by the following fields from the payload
    - number of ratings (sort_rating_number): descending
    - number of baths: ascending

- number of beds: ascending

Endpoint: /homes/owned/
Methods: GET
Fields: id, name, description, baths, beds, guests, street_address, city, state, country, image, owner, rating_sum, rating_number

Permissions:
- IsAuthenticated: Must pass JWT access token in header

Return: 200 OK and a list of homes owned by the user
Additional Notes:
- There is pagination support. Each page will display 10 results

Endpoint: /homes/<slug:pk>/edit/
Methods: PUT, PATCH
Payload: name, description, baths, beds, guests, street_address, city, state, country, image

Permissions:
- IsAuthenticated: Must pass JWT access token in header
- IsHostPermission: Must own the home with id=pk

Return: 200 OK and the updated home

Endpoint: /homes/<slug:pk>/details/
Methods: GET
Payload: id, name, description, baths, beds, guests, street_address, city, state, country, image, owner, rating_sum, rating_number

Permissions:
- None

Return: 200 OK and the home with id=pk

Endpoint: /homes/<slug:pk>/delete/
Methods: DELETE

Permissions:

- IsAuthenticated: Must pass JWT access token in header
- IsHostPermission: Must own the home with id=pk

Return: 204 No Content
Additional Notes:
- Deletes the home with id=pk, all related content will cascade and be deleted

Endpoint: /homes/<slug:pk>/add_review/
Methods: POST
Payload: rating, message

Permissions:
- IsAuthenticated: Must pass JWT access token in header
- HasStayedPermission: Must have been a renter of the home with id=pk
- NoPrevHomeComPermission: Must have *not* made a previous comment about this home with id=pk

Return: 201 Created and the newly created comment
Additional Notes:
- The home being commented about and the user making the comment are automatically passed
- The rating is added to the home's rating_sum and rating_number

Endpoint: /homes/<slug:pk>/reservations/add/
Methods: POST
Payload: id, home, renter, start_date, end_date, expiry_date, status

Permissions:
- IsAuthenticated: Must pass JWT access token in header
- isOwnerReserve: Must not be the owner of the home with id=pk

Return: 201 Created and the newly created reservation
Additional Notes:
- The user is automatically passed as the renter of the home
- The home with id=pk is automatically passed as the home of the reservations
- Before performing the create, the server validates the passed start_date, end_date, and expiry_date
    - end_date cannot be before start_date
    - start_date and expiry_date cannot be in the past
    - start_date and end_date must not overlap any other reservations of the same home that are pending, approved, or pending cancellation
    - start_date and end_date must lie within an availability range from Price

- Once the reservation is created, a new notification for the home owner is also created

Endpoint: /homes/<slug:pk>/reservations/<slug:res_pk>/edit_status/
Methods: PUT, PATCH
Payload: status

Permissions:
- IsAuthenticated: Must pass JWT access token in header
- isOwnerEdit: Must be the owner of the home with id=pk
- isReservationHome: The reservation with id=res_pk must be a reservation for the home with id=pk

Return: 200 OK and the new status
Additional Notes:
- Once it gets the reservation, it checks to see if it has expired or been completed and updates the status accordingly
- Send the renter of the reservation a notification based on the new status
- Returns a validation error if the new status is invalid
    - In other words, the current status of the reservation is not compatible with the new status

Endpoint: /homes/<slug:pk>/reservations/<slug:res_pk>/cancel/
Methods: PUT, PATCH

Permissions:
- IsAuthenticated: Must pass JWT access token in header
- isRenter: Must be the renter of the reservation with id=res_pk
- isReservationHome: The reservation with id=res_pk must be a reservation for the home with id=pk

Return: 200 OK and the new status
Additional Notes:
- Once it gets the reservation, it checks to see if it has expired or been completed and updates the status accordingly
- Sends the home owner a notification if a cancelation request is required
- The new status is automatically created and passed
- Returns a validation error if the reservation cannot be cancelled
    - The reservation has a pending cancelation request
    - The reservation has status 1, 2, 6, or 7

Endpoint: /homes/<slug:pk>/reservations/view/
Methods: GET
Fields: id, home, renter, start_date, end_date, expiry_date, status
Payload: status

Permissions:
- IsAuthenticated: Must pass JWT access token in header
- isOwnerEdit: Must be the owner of the home with id=pk

Return: 200 OK and the list of reservations for home with id=pk
Additional Notes:
- If the user passes a value for status in the GET request, the results will be filtered by this status
- Before returning the list, it makes sure there are no expired or completed reservations with statuses that do not match
- There is pagination support. Each page will display 10 results

Endpoint: /homes/<slug:pk>/availability/add/
Methods: POST
Payload: home, price, start_date, end_date,

Permissions:
- IsAuthenticated: Must pass JWT access token in header
- IsHostPermission: Must own the home with id=pk

Return: 201 Created and the new availability
Additional Notes:
- The home with id=pk is automatically passed as the home for the availability
- The creation validates that the new availability does not overlap with any other availability

Endpoint: /homes/<slug:pk>/availability/<slug:avail_id>/update/
Methods: PUT, PATCH
Payload: home, price, start_date, end_date,

Permissions:
- IsAuthenticated: Must pass JWT access token in header
- IsHostPermission: Must own the home with id=pk
- HomeMatchAvailability: The availability with id=avail_id must be for the home with id=pk

Return: 200 OK and the updated availability

Additional Notes:

- The creation validates that the new start_time and end_time for the availability does not overlap with any other availability
- For every reservation that was previously within the availability is terminated, denied, or cancelation approved, and a notification is sent

Endpoint: /homes/<slug:pk>/availability/<slug:avail_id>/delete/
Methods: DELETE

Permissions:

- IsAuthenticated: Must pass JWT access token in header
- IsHostPermission: Must own the home with id=pk
- HomeMatchAvailability: The availability with id=avail_id must be for the home with id=pk

Return: 204 No Content
Additional Notes:

- The creation validates that the new start_time and end_time for the availability does not overlap with any other availability
- For every reservation that was previously within the availability is terminated, denied, or cancelation approved, and a notification is sent

Endpoint: /homes/<slug:pk>/availability/all/
Methods: GET
Fields: home, price, start_date, end_date,

Permissions:

- None

Return: 200 OK and the availability for home with id=pk
Additional Notes:

- There is pagination support

Endpoint: /homes/<slug:pk>/images/add/
Methods: POST
Payload: home, image, alt

Permissions:

- IsAuthenticated: Must pass JWT access token in header
- IsHostPermission: Must own the home with id=pk

Return: 201 Created and the new image for home with id=pk
Additional Notes:
- The home with id=pk is automatically passed as the home for the image

Endpoint: /homes/<slug:pk>/images/<slug:image_pk>/delete/
Methods: DELETE

Permissions:
- IsAuthenticated: Must pass JWT access token in header
- IsHostPermission: Must own the home with id=pk
- ImageBelongs: Image with id=image_pk must belong to the home with id=pk

Return: 204 No Content

Endpoint: /homes/<slug:pk>/images/all/
Methods: GET
Fields: home, image, alt

Permissions:
- None

Return: 200 OK and the images for home with id=pk
Additional Notes:
- There is pagination support

<u>**Comment Views:**</u>

Endpoint: /comments/<slug:pk>/reply/
Methods: POST
Payload: message

Permissions:
- IsAuthenticated: Must pass JWT access token in header
- IsOwnedHomeComment: The comment must be about a home, and the user must be the owner of the home
- NoPreviousCommentReply: The user must not have previously replied to this comment

Return: 201 Created and the new reply for comment with id=pk
Additional Notes:
- The comment being replied to (response_object) and the commentee and the

comment (id=pk) are passed automatically, as well as the level of the response

Endpoint: /comments/response/<slug:pk>/reply/
Methods: POST
Payload: message

Permissions:

- IsAuthenticated: Must pass JWT access token in header
- IsResponseToUser: The user must be the comentee of the response with id=pk
- NoPreviousResponseReply: The user must not have previously replied to this response

Return: 201 Created and the new reply for comment with id=pk
Additional Notes:

- The comment being replied to (response_object) and the commentee and the comment (id=pk) are passed automatically, as well as the level of the response

Endpoint: /comments/<slug:pk>/update/
Methods: PUT, PATCH
Payload: rating, message

Permissions:

- IsAuthenticated: Must pass JWT access token in header
- IsCommentOwner: Must be the comentee for the comment with id=pk

Return: 200 OK and the updated comment with id=pk
Additional Notes:

- The rating_sum and rating_number of the object being commented on is updated automatically

Endpoint: /comments/response/<slug:pk>/update/
Methods: PUT, PATCH
Payload: message

Permissions:

- IsAuthenticated: Must pass JWT access token in header
- IsResponseOwner: Must be the comentee for the response with id=pk

Return: 200 OK and the updated response with id=pk

Endpoint: /comments/<slug:pk>/delete/
Methods: DELETE

Permissions:
- IsAuthenticated: Must pass JWT access token in header
- IsCommentOwner: Must be the comentee for the comment with id=pk

Return: 204 No Content

Endpoint: /comments/response/<slug:pk>/delete/
Methods: DELETE

Permissions:
- IsAuthenticated: Must pass JWT access token in header
- IsResponseOwner: Must be the comentee for the response with id=pk

Return: 204 No Content

Endpoint: /comments/user/<slug:pk>/all/
Methods: GET
Fields: rating, message

Permissions:
- None

Return: 200 OK and the comments for user with id=pk
Additional Notes:
- There is pagination support

Endpoint: /comments/home/<slug:pk>/all/
Methods: GET
Fields: id, rating, message, comentee

Permissions:
- None

Return: 200 OK and the comments for home with id=pk
Additional Notes:
- There is pagination support

Endpoint: /comments/<slug:pk>/all/
Methods: GET
Fields: level, message, comentee

Permissions:
- None

Return: 200 OK and the responses to comment with id=pk
Additional Notes:
- There is pagination support

Notification Endpoints

Endpoint: /notifications/all/
Methods: GET
Fields: id, message, notif_link, read

Permissions:
- IsAuthenticated: Must pass JWT access token in header

Return: 200 OK and the notifications for the current user
Additional Notes:
- There is pagination support

Endpoint: /notifications/<slug:pk>/read/
Methods: PUT, PATCH
Fields: id, message, notif_link, read

Permissions:
- IsAuthenticated: Must pass JWT access token in header
- OwnsNotif: Must be the notifee for the notification with id=pk

Return: 200 OK and the read notification
Additional Notes:
- The new value for read is automatically passed

Endpoint: /notifications/clear/
Methods: DELETE

Permissions:

- IsAuthenticated: Must pass JWT access token in header

Return: 204 No Content

Additional Notes:

- This view deletes all of the **read** notifications for the current user