CSC309 Group2239 P2 Content

[To run the project:]
[Entity Relationship Diagram:]
[Models Describtion:]
[Endpoints Details:]

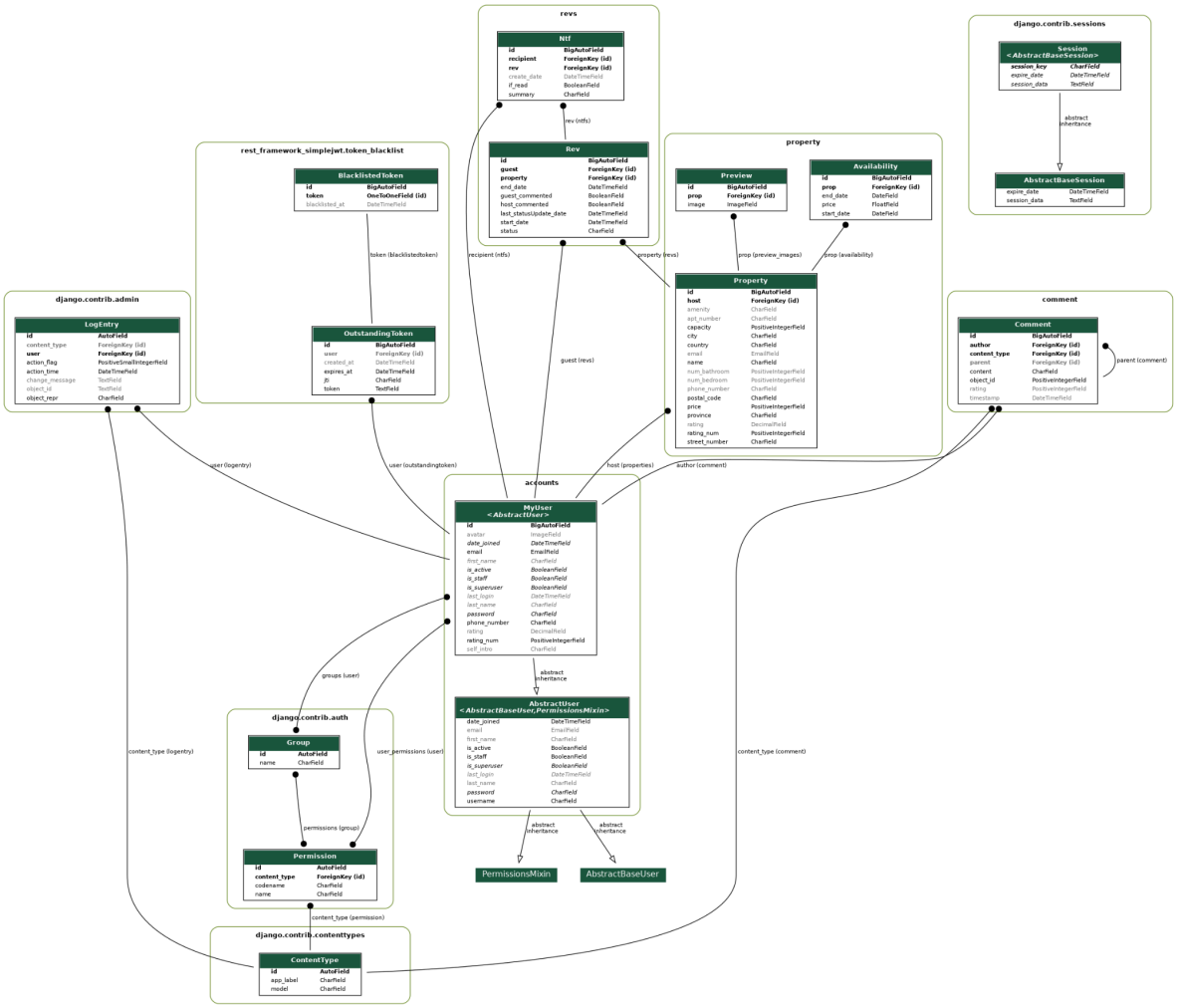
To run the project:

In project root folder:
sh startup.sh
sh run.sh

Assumptions: user has sudo rights

Note: If apt install fails, run ‘sudo apt-get update’

Entity Relationship Diagram:



Note: You can also find this diagram in our repository.

Models Describtion:

Rev Model:

```python
class Rev(models.Model):
    #    host = models.ForeignKey(User, related_name="revs", on_delete=models.CASCADE)
    guest = models.ForeignKey(settings.AUTH_USER_MODEL,
                              related_name="revs", on_delete=models.CASCADE)
    property = models.ForeignKey(
        Property, related_name="revs", on_delete=models.CASCADE)
    status = models.CharField(max_length=40, default='pending')
    # price = models.PositiveIntegerField()
    start_date = models.DateTimeField()
    end_date = models.DateTimeField()
    last_statusUpdate_date = models.DateTimeField()
    host_commented = models.BooleanField(
        default=False, blank=False, null=False)
    guest_commented = models.BooleanField(
        default=False, blank=False, null=False)

    def __str__(self):
        return f"Rev: id:{self.pk}, host: {self.property.host.email}, guest: {self.guest.email}, property: {self.property.pk}"
```

Model Rev is for reservations.

It has 2 foreign keys: guest and property. It has a field ‘status’ to keep track of the current status of the reservation (upon creation, it takes the default value ‘pending’, and its value updated whenever a reservation action is performed). It has a field ‘last_statusUpdate_date’ to store action timestamps.

An instance is created when a user makes a reservation. User needs to provide start_date and end_date. guest is the current user, property is provided in url <property_id>, status is pending by default.

Ntf Model:

```python
class Ntf(models.Model):
    recipient = models.ForeignKey(
        settings.AUTH_USER_MODEL, related_name="ntfs", on_delete=models.CASCADE)
    rev = models.ForeignKey(Rev, related_name="ntfs",
                            on_delete=models.SET_NULL, null=True)
    summary = models.CharField(max_length=200)
    if_read = models.BooleanField(default=False)
    create_date = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return f"Ntf: id:{self.pk}, recipient:{self.recipient}, rev_id:{self.rev.pk}, summary:{self.summary}, if_read:{self.if_read}"
```

Model Ntf is for notifications.

It has 2 foreign keys: ‘recipient’ and ‘rev’. It has fields ‘summary’, ‘if_read’, ‘create_date’ to store info specific to a notification instance. ‘if_read’ is false by default and will be flipped to ‘true’ once the user reads the notification (i.e. upon successful visit of the /details/ endpoint).

An instance is created upon a successful reservation action, the recipient is set to a User instance, referring to either the host or guest depending on the action.

Refer to the table for action + corresponding notification summary + recipient

Action	source status	target status	action performed by	notification msg	notification recipient
deny_reservation_request	pending	denied	host	Your reservation request is denied.	guest
approve_reservation_request	pending	approved	host	Your reservation request is approved.	guest
request_cancel_reservation	approved	pending(cancel request)	guest	You got cancel request	host
deny_cancel_request	approved	approved	host	Your cancel request is denied.	guest
approve_cancel_request	approved	canceled	host	Your cancel request is approved.	guest
terminate	approved	terminated	host	Your reservation is terminated.	guest

MyUser Model:

```
class MyUser(AbstractUser):
    username = None
    email = models.EmailField(_("email address"), unique=True)
    phone_number = models.CharField(_("phone number"), unique=True, null=True, validators=[
                                    RegexValidator(r'^\d{3}-\d{3}-\d{4}$')], max_length=20)
    avatar = models.ImageField(
        upload_to="statics/avatar/", null=True, blank=True)
    self_intro = models.CharField(
        blank=True, null=True, max_length=150, default="This guy doesn't write anything")
    rating = models.DecimalField(
        default=None, blank=True, null=True, max_digits=3, decimal_places=2)
    rating_num = models.PositiveIntegerField(default=0, null=False)
    comments = GenericRelation(Comment)

    USERNAME_FIELD = "email"
    REQUIRED_FIELDS = []

    objects = MyUserManager()

    def __str__(self):
        return self.email
```

The MyUser Model is used for the user account system. A normal user should be registered with a unique email, a unique phone number, first_name and

last_name. Users can also choose to upload images as their avatar and provide a short self introduction which will be displayed on their profile page.

The format of the phone number field should be xxx-xxx-xxxx. Otherwise the registration will not succeed.

Rating and rating_num can not be modified by users themselves. Instead, whenever there is a new rating submitted by a host, these fields will be updated.

Comment Model:

```
class Comment(models.Model):
    content_type = models.ForeignKey(ContentType, on_delete=models.CASCADE)
    author = models.ForeignKey(
        settings.AUTH_USER_MODEL, default=1, null=True, on_delete=models.SET_NULL)
    object_id = models.PositiveIntegerField()
    content_object = GenericForeignKey('content_type', 'object_id')
    content = models.CharField(max_length=200, blank=False, null=False)
    rating = models.PositiveIntegerField(blank=True, null=True)
    parent = models.ForeignKey(
        "self",  blank=True, null=True, on_delete=models.CASCADE)
    timestamp = models.DateTimeField(auto_now_add=True)

    class Meta:
        ordering = ['-timestamp']

    def children(self):
        return Comment.objects.filter(parent=self)

    @property
    def is_parent(self):
        if self.parent is not None:
            return False
        return True
```

The comment model utilizes the Django ContentType model. Allow itself binds to multiple types of models. Therefore this model is responsible for both user comment and property comment. Comments with no parent are considered as the primary comment. Those with parents are considered replies to their parent comment.

To create the comment, authentication and content are required. Rating is optional. However children comments/replies can not be created with ratings. On display, comments will be shown in a newest to oldest order.

Property Model:

```
# Create your models here.
class Property(models.Model):
    name = models.CharField(max_length=50)
    country = models.CharField(max_length=50)
    province = models.CharField(max_length=50)
    city = models.CharField(max_length=50)
    apt_number = models.CharField(max_length=20, blank=True, null=True)
    street_number = models.CharField(max_length=20)
    postal_code = models.CharField(max_length=20)

    email = models.EmailField()
    phone_number = models.CharField(_("phone number"), validators=[
                                    RegexValidator(r'^\d{3}-\d{3}-\d{4}$')], max_length=20)

    price = models.DecimalField(
        _(u'Price'), decimal_places=2, max_digits=12,
        validators=[MinValueValidator(Decimal('0.01'))])
    rating = models.DecimalField(
        default=None, blank=True, null=True, max_digits=3, decimal_places=2)
    rating_num = models.PositiveIntegerField(default=0, null=False)
    amenity = models.CharField(max_length=255, blank=True, default=0)

    num_bedroom = models.PositiveIntegerField(blank=True, default=0)
    num_bathroom = models.PositiveIntegerField(blank=True, default=0)
    capacity = models.PositiveIntegerField()

    host = models.ForeignKey(settings.AUTH_USER_MODEL, on_delete=models.CASCADE,
                             related_name='properties')
    comments = GenericRelation(Comment)
```

This model is for storing properties. It has a name field, some address fields (apt # can be left blank for property that are not in an apartment), a default price of the property, the host can specify another price when creating availability for this property. Email and phone_number can be left as blank when creating and updating the property, in that case the host's email and phone_number will be put into those fields. Amenity is stored as a comma-separated strings to store multiple amenities.

Preview Model:

```
class Preview(models.Model):
    image = models.ImageField(upload_to='property/')
    prop = models.ForeignKey(Property, on_delete=models.CASCADE,
                             related_name='preview_images')
```

This model stores each preview image of the property, they will all be deleted when the property is deleted.

Availablilty Model:

```
class Availability(models.Model):
    start_date = models.DateField()
    end_date = models.DateField()
    price = models.DecimalField(
        _(u'Price'), decimal_places=2, max_digits=12,
        validators=[MinValueValidator(Decimal('0.01'))])
    prop = models.ForeignKey(Property, on_delete=models.CASCADE,
                             related_name='availability')
```

This model stores each availability of the property. It has fields start_date and end_date to specify the time range of this availability. It also has a price field which can be left as blank, in that case, the property's default price will be put to the field. When creating and updating availability, no overlap is allowed for a single property.

Endpoints Details:

Accounts:

All of the following endpoints require authentication/login except Register and Login.

Register:

Endpoint: /accounts/register/

Methods: POST

Fields/payload: email, phone_number, password1, password2, first_name, last_name, avatar, self_intro

Validation error: All fields except avatar and self_intro are mandatory and cannot be left blank. If so, the following error must be shown above the problematic field:

- This field is required.

Other errors:

- Password fields didn't match.
- Enter a valid email address.
- Enter a valid value.(for phone number)

Additional notes: The format of Phone number should be xxx-xxx-xxxx.

Login:

Endpoint: /accounts/login/

Methods: POST

Fields/payload: email, password

Invalid inputs:

If the credentials are wrong, the follow will show:

- No active account found with the given credentials

If the credentials are missing, the follow will show:

- This field may not be blank.

Additional notes: On success, users should get valid tokens for authentication provided by SimpleJWT.

Logout:

Since we are using token based authentication, logout would simply be delete the token from the browser or front end. Which will be taken care of in P3.

Update:

Endpoint: /accounts/profile/update/

Methods: PUT,PATCH

Fields/payload: email, phone_number, password1, password2, first_name, last_name, avatar, self_intro

Validation errors:

- Password fields didn't match.
- Enter a valid email address.
- Enter a valid value.(for phone number)

View own profile:

Endpoint: /accounts/home/

Methods: GET

Fields/payload: email, id, phone_number, first_name, last_name, avatar, self_intro, rating, rating_num

Additional notes: This endpoint will show the authenticated user's own profile.

View guest profile:

Endpoint: /accounts/<pk>/profile/

Methods: GET

Fields/payload: email, id, phone_number, first_name, last_name, avatar, self_intro, rating, rating_num

Additional notes: This endpoint will show the authenticated user the profile of the user whose id is pk.

Comment:

All of the following endpoints require authentication/login. Users can not create parent comments on themselves or their own properties(will return a 403 response).

Create User Comment:

Endpoint: /comment/create/myuser/<int:user_id>/<int:parent_id>/

Methods: POST

Fields/payload: rating, content

Validation error:

- Rating can not larger than 5

Additional notes: A host can only comment on a user who has a completed/terminated reservation with the host. The limit is that each reservation can only be commented once(excluding the replies).

Create Property Comment:

Endpoint: /comment/property/<int:property_id>/<int:parent_id>/

Methods: POST

Fields/payload: rating, content

Validation errors:

- Rating can not larger than 5

Additional notes: A user can only comment on property with a completed/terminated reservation. The limit is that each reservation can only be commented once(excluding the replies).

Replies to Comment:

Endpoint: /comment/update/<int:pk>/

Methods: PUT,PATCH

Fields/payload: content

Validation errors:

- Password fields didn't match.
- Enter a valid email address.
- Enter a valid value.(for phone number)

View User Comment:

Endpoint: list/user/<int:pk>/

Methods: GET

Additional notes: This endpoint will show all the comments of the user with id = pk.

View Property Comment:

Endpoint: list/property/<int:pk>/

Methods: GET

Additional notes: This endpoint will show all the comments of the property with id = pk.

Edit Comment:

Endpoint: /comment/update/<int:pk>/

Methods: PUT

Fields/payload: content, rating

Additional notes: This endpoint will update the comment with id=ok. Only comments that have no parent can edit the rating.

Property:

Property index page:

Endpoint: /property/index/

Query: search, amenity, start_date, end_date, num_guest, ordering

Methods: GET

Fields/payload: id, availability, preview, name, country, province, city, apt_number, street_number, postal_code, email, phone_number, price, rating, rating_num, amenity, num_bedroom, num_bathroom, capacity, host

Additional notes:

- Does not require login
- Enter location in the search field
- Support filter by location (including country, province, city), amenity, start_date, end_date, num_guest
- Support ordering by rating, rating_num
- Pagination is supported, only 5 properties will be displayed on single page
- User may enter start_date that is later than the end_date, but no result will be shown

Property Detail page:

Endpoint: /property/<int:pk>/detail/

Methods: GET

Fields/payload: id, availability, preview, name, country, province, city, apt_number, street_number, postal_code, email, phone_number, price, rating, rating_num, amenity, num_bedroom, num_bathroom, capacity, host

Additional notes:

- See the detail of the property

Add Property:

Endpoint: /property/add/

Methods: POST

Fields/payload: name, country, province, city, apt_number, street_number, postal_code, email, phone_number, price, amenity, num_bedroom, num_bathroom, capacity, previews

Validation error:

- name, country, province, city, street_number, postal_code, price, amenity, num_bedroom, num_bathroom, capacity are mandatory and cannot be left blank. If so, the following error must be shown above the problematic field: This field is required
- price, num_bedroom, num_bathroom, capacity must be positive and num_bedroom, num_bathroom, capacity must be integer; otherwise corresponding error message will come up

Additional notes:

- Requires login
- apt_number, email, phone_number, amenity, num_bedroom, num_bathroom, previews are not required fields.
- When email or phone_number is not input, the host's email or phone_number will be stored

Update Property:

Endpoint: /property/<int:pk>/update/

Methods: PUT

Fields/payload: name, country, province, city, apt_number, street_number, postal_code, email, phone_number, price, amenity, num_bedroom, num_bathroom, capacity, previews

Validation error:

- name, country, province, city, street_number, postal_code, price, amenity, num_bedroom, num_bathroom, capacity are mandatory and cannot be left blank. If so, the following error must be shown above the problematic field: This field is required

- price, num_bedroom, num_bathroom, capacity must be positive and num_bedroom, num_bathroom, capacity must be integer; otherwise corresponding error message will come up

Additional notes:

- Requires login
- apt_number, email, phone_number, amenity, num_bedroom, num_bathroom, previews are not required fields.
- When email or phone_number is not input, the host's email or phone_number will be stored
- Return 404 NOT FOUND if pk is not an id of Property
- Return 403 FORBIDDEN if the logged in user is not the owner of the property

Delete Property:

Endpoint: /property/<int:pk>/delete/

Methods: DELETE

Additional notes:

- Requires token
- Return 404 NOT FOUND if pk is not an id of Property
- Return 403 FORBIDDEN if the logged in user is not the owner of the property

Delete Preview:

Description: delete a preview photo of a property

Endpoint: /property/preview/<int:pk>/delete/

Methods: DELETE

Additional notes:

- Requires token
- Return 404 NOT FOUND if pk is not an id of Preview
- Return 403 FORBIDDEN if the logged in user is not the owner of the property

Add Availability:

Endpoint: /property/<int:prop_pk>/availability/add/

Methods: POST

Fields/payload: start_date, end_date, price

Validation error:

- All fields are mandatory and cannot be left blank. If so, the following error must be shown above the problematic field: This field is required
- start_date cannot be later than end_date
- [start_date, end_date] must not overlap with other availabilities
- price must be positive

Additional notes:

- Requires login
- Return 404 NOT FOUND if prop_pk is not an id of Property
- Return 403 FORBIDDEN if the logged in user is not the owner of the property

Update Availability:

Endpoint: /property/availability/<int:pk>/update/

Methods: PUT

Fields/payload: start_date, end_date, price

Validation error:

- All fields are mandatory and cannot be left blank. If so, the following error must be shown above the problematic field: This field is required
- start_date cannot be later than end_date
- [start_date, end_date] must not overlap with other availabilities
- price must be positive

Additional notes:

- Requires login
- Return 404 NOT FOUND if prop_pk is not an id of Property
- Return 403 FORBIDDEN if the logged in user is not the owner of the property

Delete Availability:

Endpoint: /property/availability/<int:pk>/delete/

Methods: DELETE

Additional notes:

- Requires token
- Return 404 NOT FOUND if pk is not an id of Availability
- Return 403 FORBIDDEN if the logged in user is not the owner of the property of this availability

Reservations:

All of the following endpoints require authentication/login except Register and Login.

List:

Endpoint: /revs/list/<user_id>/?user_type=&status=

Description: Displays a list of <user_id>'s reservations.

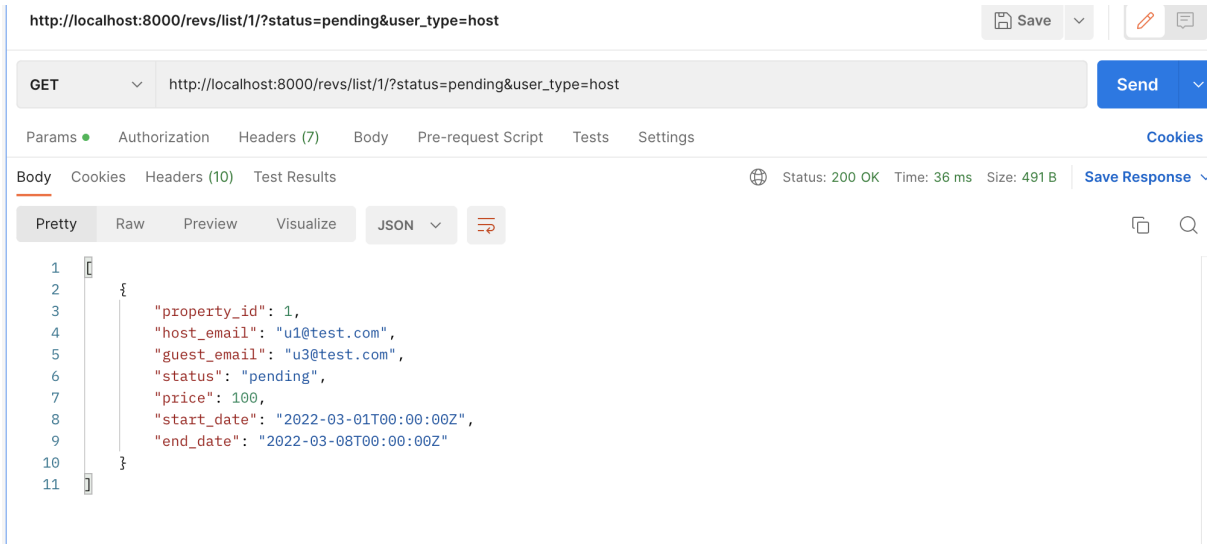
Optional url query param (user_type, status) as filters:

- user_type: host, guest
- status: pending, denied, approved, canceled, terminated, completed, pending(cancel request), expired

Methods: GET

Fields/payload: N/A

Example response:



Validation errors:

- 403: cannot review reservation of another user
- 404: invalid filter value

Note: if using filter, endpoint no trailing /

Action:

Reserve:

Endpoint: /revs/create/<pid>/

Description: - Creates a reservation of <pid>, payload start_date,end_date.

- Upon success creation, returns a summary of the reservation & notifies the host of the pending request.

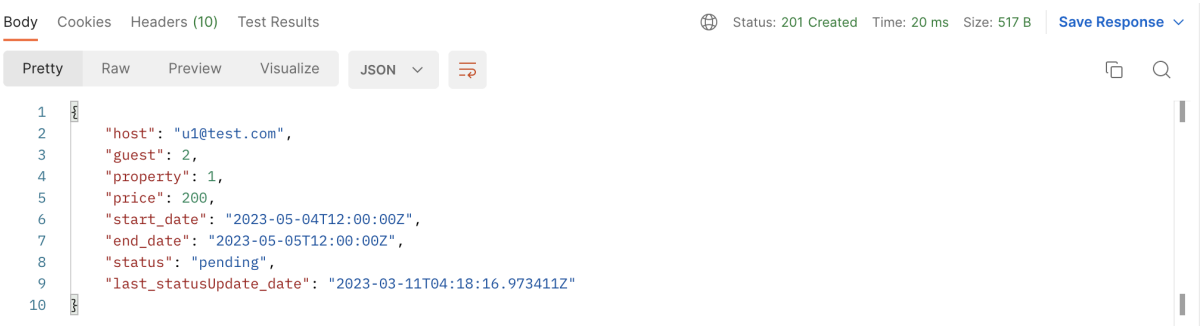
Methods: POST

Fields/payload: start_date, end_date

Example Request Body: 2023-05-04T12:00:00Z

	KEY	VALUE
☑	start_date	2023-05-04T12:00:00Z
☑	end_date	2023-05-05T12:00:00Z

Example Response



Validation errors:

- 404: invalid start_date and end_date:missing or start>=end
- 404: <property_id> not exists
- 400: start_date and end_date overlaps with existing reservations of same property
- 400: check potential guest is not host

All other transition actions

Endpoint: /revs/action/<reservation_id>/<action>/

Description: - Update status of <reservation_id> according to <action>, upon success update, return current status with timestamp of the action, notify user (host or guest depending on the action)

- Before update, check <action> consistent with current user's type and current reservation status.

Action	source status	target status	action performed by	notification msg	notification recipient
deny_reservation_request	pending	denied	host	Your reservation request is denied.	guest
approve_reservation_request	pending	approved	host	Your reservation request is approved.	guest
request_cancel_reservation	approved	pending(cancel request)	guest	You got cancel request	host
deny_cancel_request	approved	approved	host	Your cancel request is denied.	guest
approve_cancel_request	approved	canceled	host	Your cancel request is approved.	guest
terminate	approved	terminated	host	Your reservation is terminated.	guest

Fields/payload: N/A

Validation errors:

- 403: current user is neither host nor guest of this specific reservation
- 404: action and current usre_type not consistent
- 400: action not defined

Notifications:

All of the following endpoints require authentication/login except Register and Login.

List:

Endpoint: /notifications/<user_id>/list/

Description: Displays a list of <user_id>'s notifications

Methods: GET

Fields/payload: N/A

Example Response:


```
{
    "count": 1,
    "next": null,
    "previous": null,
    "results": [
        {
            "pk": 1,
            "rev_id": 1,
            "recipient_email": "u1@test.com",
            "summary": "New Pending Reservation Request",
            "if_read": true,
            "create_date": "2023-03-11T04:14:22.289997Z",
            "guest_email": "u2@test.com",
            "host_email": "u1@test.com"
        }
    ]
}
```


Validation errors:

- 403: cannot view other user's notification list

Read notification:

Endpoint: /notifications/<user_id>/<notification_id>/details/

Description: - Displays <user_id>'s <notification_id>.

- Upon success, set notification if_read to true

Method: GET

Fields/payload: N/A

Example Response:

```
{
    "count": 1,
    "next": null,
    "previous": null,
    "results": [
        {
            "pk": 1,
            "rev_id": 1,
            "recipient_email": "u1@test.com",
            "summary": "New Pending Reservation Request",
            "if_read": true,
            "create_date": "2023-03-11T04:14:22.289997Z",
            "guest_email": "u2@test.com",
            "host_email": "u1@test.com"
        }
    ]
}
```


Validation errors:

- 403: cannot view other user's notification details
- 404: <notification_id> not exists

Clear Notification: each instance of notification has a field 'if_read', default is false, set to true after valid visit of /notifications/<user_id>/<notification_id>/details/

Receiving Notification: notifications are sent/created upon success reservation actions

i.e. host get notified upon new (1) reservation and (2) cancel request,

guest get notified upon (3) approve (4) cancel of their requests