P2

User Section

- **user/register/**
  - Creates a new user with fields
- **user/profile/**
  - User checks their own profile
  - Must be authenticated
- **user/profile/edit/**
  - User edits their own profile
  - Must be authenticated
- **user/login/**
  - User logs in with an access token
- **user/login/refresh/**
  - User refreshes token
  - Must be authenticated
- **user/logout/**
  - User logout

User Model

RestifyUser

- phone: This field is a CharField with a max_length of 11 characters, meaning it can store a phone number up to 11 digits long. It is also marked as blank=True, which means it can be left empty if the user doesn't want to provide a phone number.
- contact_method: This field is also a CharField, but it has a limited set of choices defined in the CONTACT_CHOICES list. The max_length is set to 5, which is enough to store either 'phone' or 'email'. By default, the value will be 'phone' if not specified otherwise.
- profile_image: This field is an ImageField, which means it stores an image file. It is marked as null=True, which means it can be left empty if the user doesn't want to upload a profile image. The upload_to argument is set to "profile_pictures/", which specifies the subdirectory where uploaded images will be stored.

Comment Section

- **comment/property/<str:property_id>/add/**
  - Guest create a comment & rating or reply to a property
  - Host can reply to the guest comment
  - Guest can reply to the Host reply etc.
  - The rating will be calculated by average of all comments (not reply) of this property and update the property rating
  - The host or the guest will be notified if they received a comment & rating or reply
  - Error handling & permission:
    Must be authenticated
    - Comments

        - No empty text
        - No empty/invalid rating (0 to 5)
        - Guest can only rate and comment on a property they had ‘terminated’ or ‘completed’ at most once
    - Replies
        - No empty text
        - They cannot reply to themself
        - Must have a valid comment (parent) to respond to
        - Cannot have a rating for a reply
- **comment/property/<str:property_id>/**
    - Shows all comments of this property.
    - No authentication required
- **comment/property/<str:property_id>/<int:pk>/**
    - Shows the details of comment pk
    - No authentication required
- **comment/property/<str:property_id>/guest/<str:guest_id>/add/**
    - The host creates a comment & rating a guest.
    - The guest will be notified if they received a comment & rating
    - Error handling & permission:
      Must be authenticated
        - Only the host of the property can comment
        - Only one comment
        - Guest must have completed a reservation
        - No empty text
        - No empty/invalid rating (0 to 5)
- **comment/property/<str:property_id>/guest/**
    - Any user that has properties are hosts and only hosts can view the comments & rating of all guest on this property.
    - Must be authenticated.

Comment Models

Comment

- commenter: This field is a ForeignKey to the RestifyUser model, which means it stores a reference to a user who left a comment. The on_delete argument is set to models.CASCADE, which means that if the user is deleted, all their associated comments will also be deleted. This field is marked as null=False and blank=False, which means it is required and must have a value.
- property: This field is a ForeignKey to the Property model, which means it stores a reference to the property that this comment is about. The on_delete argument is set to models.CASCADE, which means that if the property is deleted, all associated comments

will also be deleted. This field is also marked as null=False and blank=False, which means it is required and must have a value.

- added_date: This field is a DateTimeField with auto_now_add=True, which means it automatically gets the current date and time when the comment is created.
- text: This field is a CharField with a max_length of 300 characters, which means it can store a comment up to 300 characters long. This field is marked as null=False and blank=False, which means it is required and must have a value.
- rating: This field is an IntegerField with validators to ensure the value is between 0 and 5. It is also marked as null=True and blank=True, which means it can be left empty if the comment is a reply (replies cannot have a rating).
- parent: This field is a ForeignKey to the self model, which means it stores a reference to another comment that this comment is a reply to. This field is marked as null=True and blank=True, which means it can be left empty if the comment is not a reply. The help_text argument provides additional information about this field.
- Meta: This inner class provides additional metadata for the model. The ordering attribute specifies that comments should be ordered by their added_date, with the most recent comments first.
- __str__: This method returns a string representation of the comment, which includes the comment's ID, the name of the property it relates to, and the username of the commenter.
- children: This method returns all comments that are replies to this comment.
- is_parent: This method returns a boolean indicating whether this comment is a parent comment (i.e. not a reply).

HostToGuestComment

- host_commenter: ForeignKey to RestifyUser with related_name="host_commenter". The user who is the host and left the comment.
- guest: ForeignKey to RestifyUser with related_name="guest". The user who is the guest and is being reviewed.
- property: ForeignKey to Property. The property that the guest stayed in.
- added_date: DateTimeField with auto_now_add=True. The date and time when the comment is created.
- text: CharField with max_length=300. The comment text.
- rating: IntegerField with validators for 0-5. Rating for the guest.

Notification Section

- **notification/all/**
    - Must be authenticated
    - Show all notification messages that the current logged-in user receives
    - Method: GET
- **notification/<int:pk>/detail/**
    - Must be authenticated
    - Providing the message id, user can mark the particular messages as read

    - User can delete the message by using the DELETE method
    - Method: GET, DELETE
    - If we cannot find the pk as the id for a notification message, it will return 404_NOT_FOUND
    - If people use the DELETE method, and once the message is deleted successfully, it will return 204_NO_CONTENT

Notification model

- user : a foreign key field to the RestifyUser model, representing the user who owns the notification message.
- date : a DateTimeField that stores the date and time when the notification was created. The auto_now_add argument ensures that the field is automatically set to the current date and time when the object is first created.
- title: a CharField that stores the title of the notification. It has a maximum length of 100 characters and can be blank or have a default value of an empty string.
- read: a BooleanField that indicates whether the notification has been read. It has a default value of False.
- text: a TextField that stores the content of the notification.
- notification_type: a CharField that stores the type of the notification. It has a maximum length of 20 characters and uses the NOTIFICATION_TYPE_CHOICES constant to specify the available choices. The default value is set to 'reservation'

NOTIFICATION_TYPE_CHOICES can be either ‘reservation’ or ‘cancellation’ or ‘approval’

Reservation Section

- **reservation/all/**
    - Must be authenticated
    - **Method**: GET
    - Description
        - Show all reservation records for the logged-in user, either the user is the host or a guest
- **reservation/all/<str:criteria>/**
    - Must be authenticated
    - **Method**: GET/DELETE
    - **Description**
        - If we cannot find any reservation records for the current logged_in user or the results after applying filters, 204_NO_CONTENT will be returned
        - For the logged-in user, the url can show reservation records be filtered by user type(host/guest)

        - For the logged-in user, the url can also show reservation records be filtered by reservation status (pending, denied, expired, approved, canceled, terminated, completed)
        - Therefore, the **<str:criteria>** can either be user type or reservation status

* In the reservation model, we will check the check-in date and check-out date to see if the reservation duration has passed the passed the current date or not, if the reservation happens in the past, the reservation_status will change to ‘expired’

- **reservation/create**
    - Must be authenticated
    - **Method**: POST
    - **Payload**:The request data should be in JSON format and contain fields for check_in, check_out, host, liable_guest,number_of_guests,reservations_status
    - **Description**
        - If the post form data cannot pass serializer validation, 400_BAD_REQUEST will be returned
        - Only guest can makes request to reserve a property, we will set the logged-in user as the guest in the reservation request
        - When the logged-in user makes reservation successfully, it will send a notification to property host
- **reservation/<int:pk>/approved/**
    - Must be authenticated
    - **<int:pk>** is the reservation_id
    - **Method**: PATCH
    - **Payload**:The request data should be in JSON format and contain fields for check_in, check_out, host, liable_guest,number_of_guests,reservations_status
    - **Description**
        - If the PATCH data cannot pass serializer validation, 400_BAD_REQUEST will be returned, else 200 will be returned
        - If the current logged_in user is the host, the reservation has already been submitted by user, and the reservation_status is ‘pending’, host can approve the reservation request and change reservation_status to 'approved'
        - Once the host approved the request, it will send a notification message to guest
        - The property available date is updated
- **reservation/<int:pk>/cancel/**
    - Must be authenticated
    - **<int:pk>** is the reservation_id
    - **Method**: PATCH

    - **Payload**:The request data should be in JSON format and contain fields for check_in, check_out, host, liable_guest,number_of_guests,reservations_status
    - **Description**
        - If the PATCH data cannot pass serializer validation, 400_BAD_REQUEST will be returned, else 200 will be returned
        - The view will check logged-in user is host or liable_guest of the reservation record
            - If the logged-in user is the liable_guest for the reservation request, and the reservation has already been approved by property host before (which means that the reservation status is ‘approved’), user can cancel the reservation request and the reservation status changes to 'canceled’
                - The notification message will send to host
                - The property available dates will also been updated
            - If the logged_in user is the host, the reservation is submmited by user and the reservation_status is pending, host can cancel the reservation request, and the reservation status changes to 'denied'
                - The notification message will send to guest
                - The property available dates will also been updated
            - If the logged_in user is the host, and the reservation has already been approved by host before (which means that the reservation status is ‘approved’), host can still cancel the reservation request and the reservation status changes to 'terminated'
                - The notification message will send to guest
                - The property available dates will also been updated
- **reservation/<int:pk>/completed/**
    - Must be authenticated
    - **<int:pk>** is the reservation_id
    - **Method**: PATCH
    - **Payload**:The request data should be in JSON format and contain fields for check_in, check_out, host, liable_guest,number_of_guests,reservations_status
    - **Description**
        - If the PATCH data cannot pass serializer validation, 400_BAD_REQUEST will be returned, else 200 will be returned
        - If the logged_in user is the host or liable_guest, the reservation_status has been approved by host before and the reservation end date has passed the current date, host or liable_guest can complete the reservation request and change reservation_status to 'completed'

Reservation model

- check_in: a DateField that stores the check-in date for the reservation.

- check_out: a DateField that stores the check-out date for the reservation.
- host: a ForeignKey field that refers to the RestifyUser model and indicates the user who is hosting the reservation.
- liable_guest: a ForeignKey field that refers to the RestifyUser model and indicates the user who is responsible for the reservation.
- place: a ForeignKey field that refers to the Property model and indicates the property that is being reserved. This field is optional, as indicated by the blank=True and null=True arguments.
- number_of_guests: a PositiveIntegerField that stores the number of guests for the reservation. It has a default value of 0.
- _reservation_status: a CharField that stores the current status of the reservation. It has a maximum length of 25 characters and uses the RES_STATUS constant to specify the available choices. The default value is set to 'pending', and the db_column argument specifies the name of the column in the database that stores the value.

Property Section

- **property/create/**
    - Must be authenticated
    - **URL**: /api/properties/create/
    - **HTTP Method**: POST
    - **Payload**:The request data should be in JSON format and contain fields for name, description, address, and default_price.
    - **Description**:Creates a new property with the specified details, sets the authenticated user as the owner, and generates a list of availability for the property based on the number of days in each month. Amenity can be multiple. Location has to be two capital letter Canada province.
- **property/delete/<int:pk>/**
    - Must be authenticated
    - **HTTP Method**: DELETE
    - **Payload**: The request should not contain any payload.
    - **Description**: Deletes the property with the specified ID, but only if the authenticated user is the owner of the property. If the user is not the owner, a 404 error is returned.
- **property/update/<int:pk>/**
    - Must be authenticated
    - **HTTP Method**: PUT/PATCH
    - **Query Parameters**
    - start_date (string): The start date of the availability range to update (required if end_date is provided).

    - end_date (string): The end date of the availability range to update (required if start_date is provided).
    - start_month (integer): The start month of the availability range to update (required if end_month is provided).
    - end_month (integer): The end month of the availability range to update (required if start_month is provided).
    - none (string): If set to "true", sets the availability of the specified range to None.
    - price (integer): If provided, sets the availability of the specified range to the specified price.
- **property/search/**
    - **HTTP Method**: GET
    - **Query Parameters**
    - location (string): The location of the property to search for.
    - rating (float): The minimum rating of the property to search for.
    - guest_capacity (integer): The minimum guest capacity of the property to search for.
    - amenity (list): A list of amenities that the property must have.
    - order_by (string): The field to order the search results by (price or rating).
    - start_date (string): The start date of the availability range to search for.
    - end_date (string): The end date of the availability range to search for.
    - start_month (integer): The start month of the availability range to search for.
    - end_month (integer): The end month of the availability range to search for.
    - **Description**:
    - Searches for properties based on the provided query parameters. The view applies filters to location, rating, guest capacity, and amenities. It also accepts query parameters to search for available properties within a given range of dates. The view then applies ordering based on the specified field (price or rating). The search results are returned as a list of serialized property objects.

Property model:

- id: An auto-incrementing primary key field to uniquely identify each property.
- property_name: A character field to store the name of the property. This field has a maximum length of 200 characters.
- description: A text field to store a description of the property.
- owner: A foreign key field to the RestifyUser model, representing the user who owns the property. This field is set to null=True and blank=True, meaning that a property can exist without an owner assigned to it.

- property_images: A JSON field to store a list of image URLs representing the property.
- default_price: An integer field to store the default price of the property.
- location: A character field with a maximum length of 2 to store the location of the property. This field uses a pre-defined set of choices representing different Canadian provinces and territories.
- available_dates: A JSON field to store a list of available dates for the property.
- guest_capacity: An integer field to store the maximum number of guests that the property can accommodate.
- amenity: A MultiSelectField to store a list of amenities available at the property. This field uses a pre-defined set of choices representing different amenities like pool, hot tub, gym, etc. A MaxValueMultiFieldValidator is used to ensure that no more than 3 amenities can be selected at once.
- rating: An integer field to store the rating of the property, with choices ranging from 0 to 5.