Models

CustomUser
Fields:

- Email
- First_name (not required)
- Last_name (not required)
- Phone_num (not required)
- Avatar (not required)
- Is_active (default =True)
- Is_staff (default =False)
- is_superuser (default =False)
- NOTE: USERNAME_FIELD is replaced with email
- NOTE: password is not a field on model but is a given required

Notification
Fields:

- Content (the content of the message)
- Belongs_to (foreign key to customuser)
- Cleared (default = false)

Comment
Fields:

- from_user (foreign key to CustomUser)
- content
- reply_to (foreign key to self, not required)

GuestComment(Comment)
Fields:

- guest (foreign key to CustomUser)

PropertyComment(Comment)
Fields:

- property (foreign key to Property)

Property
Fields:

- title
- description
- owner(foreign key to custom user)
- location
- num_bed
- num_bath
- num_guests
- price
- amenities (many to many to Amenities, not required)

Image
Fields:

- image (image field)
- my_property (foreign key to Property)

Reservation
Fields:

- user (foreign key to CustomUser)
- property (foreign key to Property)
- status (default='pending')
- start_date
- end_date

API Endpoints

Login

Endpoint: http://localhost:8000/api/token/

Description: if an account exists, function logs user in and returns refresh and access token. If the account doesn't exist, it returns a 401 unauthorized

Methods: POST

Body:

- Email
- password

Example Responses:

On success:

200OK


```
{
   "refresh":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ0b2tlbl90eXBlIjoicmVmcmVzaCIsImV4cCI6MTY3OTAxOTkzMi
wiaWF0IjoxNjc4OTMzNTMyLCJqdGkiOiI0YThmODc0MzM5Yjc0NTQ1OWNjYzA3M2VlNTk5Zjk2NCIsInVzZXJfaWQiOj
l9.HCng-s77uEhJo4SGT2dNPFnM1A1IT1fvpaaTQQ3miWI",
   "access":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ0b2tlbl90eXBlIjoiYWNjZXNzIiwiZXhwIjoxNjc5MDE5OTMyLC
JpYXQiOjE2Nzg5MzM1MzIsImp0aSI6IjlmZDZmNTU1ODFhYjQyYTI5MjI3ODM2OWI3MGMzNWVmIiwidXNlcl9pZCI6OX
0.xrqslt0YLKg30GHZR2a2CxTMvY0fIv_IYa8ccNwUJ1I"
}
```


If account not found:

401Unauthorized


```
{
   "detail": "No active account found with the given credentials"
}
```


Logout

Endpoint: http://localhost:8000/accounts/logout/

Description: if account is authenticated, logs user out and returns 204, otherwise returns different errors, depending on the issue (described later on)

Methods: POST

Header: must add access token as Authorization

Body:

- Refresh_token (the refresh token given by login API response

Example Responses:

On success:

```
204 No Content
```

If Authorization not provided:

401Unauthorized

```
{
   "detail": "Authentication credentials were not provided."
}
```

If incorrect access or refresh token:

401Unauthorized

```
{
   "detail": "Token is invalid or expired"
}
```

If account already has been logged out:

401Unauthorized

```
{
   "detail": "Token is blacklisted"
}
```

Signup

Endpoint: http://localhost:8000/accounts/signup/

Description: signs user up and returns 201 if mandatory fields are provided, email is valid, and not used, password is 8 characters long, not too common, and not only numeric. Otherwise, returns different errors

Methods: POST

Body:

- Email

- Password
- Password2
- First_name (not required)
- Last_name (not required)
- Phone_num (not required)
- Avatar (not required, file upload)

Example Responses:

On success:

201Created


```
{
   "email": "michaelscott@test.com",
   "password":
"pbkdf2_sha256$390000$w9ONzEDy4K4TMEZ0zOHd7G$JGv3B+yHIWDSPTDazA7xAROJko5V9xHJqlFM63ZQbnY=",
   "first_name": null,
   "last_name": null,
   "phone_num": null,
   "avatar": null
}
```


If email invalid:

400Bad Request


```
{
   "email": [
       "Enter a valid email address."
   ]
}
```


If email already in use:

400Bad Request


```
{
   "email": [
       "custom user with this email already exists."
   ]
}
```


If password is too short:

400Bad Request

```
{
   "non_field_errors": [
       "This password is too short. It must contain at least 8 characters."
   ]
}
```

Password too common:

400Bad Request

```
{
   "non_field_errors": [
       "This password is too common."
   ]
}
```

Profile view and Edit

Endpoint: http://localhost:8000/accounts/profile/<int:pk>/

Description: view profile if a get request, edit profile if a post request

Methods: GET, PUT

Header: must add access token as Authorization

Body (for PUT):

- NOTE: none of the fields are required
- Email
- Password
- Password2
- First_name
- Last_name
- Phone_num
- Avatar (file upload)

Example Responses:

On success for GET:

200OK


```
{
   "phone_num": null,
   "first_name": "",
   "last_name": "",
   "email": "michelle10@test.com",
   "avatar": null
}
```


On success for PUT:

200OK


```
{
   "phone_num": null,
   "first_name": "test",
   "last_name": "test",
   "email": "michelle10@test.com",
   "avatar": null
}
```


If logged in user is trying to view or edit a different user's profile:

400Bad Request


```
{
   "authorize": "You dont have permission for this user."
}
```


If user not authorized:

401Unauthorized


```
{
   "detail": "Authentication credentials were not provided."
}
```


Notification List

Endpoint: http://localhost:8000/accounts/notification/list/

Description: shows notifications for logged in user that have not been cleared (a notification is cleared after viewing)

Methods: GET

Header: must add access token as Authorization

Example Responses:

On success:

200OK


```
{
   "count": 5,
   "next": "http://localhost:8000/accounts/notification/list/?page=2",
   "previous": null,
   "results": [
       {
           "content": "reservation was approved for property 2",
           "belongs_to": 9,
           "cleared": false
       },
       {
           "content": "reservation was canccelled by host for proerty 2.",
           "belongs_to": 9,
           "cleared": false
       },
       {
           "content": "another notif",
           "belongs_to": 9,
           "cleared": false
       },
       {
           "content": "another notif",
           "belongs_to": 9,
           "cleared": false
       }
   ]
}
```


If user not authorized:

401Unauthorized


```
{
   "detail": "Authentication credentials were not provided."
}
```

Notification View

Endpoint: http://localhost:8000/accounts/notification/<int:pk>/

Description: view a specific notification of logged in user, viewing notification will change field cleared to

True

Methods: GET

Header: must add access token as Authorization

Example Responses:

On success:

200OK


```
{
   "content": "reservation was canceled by host for property 2.",
   "belongs_to": 9,
   "cleared": true
}
```


If account not of logged in user:

400Bad Request


```
{
   "authorize": "You dont have permission for this user."
}
```


Property Search

Endpoint: http://localhost:8000/properties/search/

Parameters:

- num_guests, number, optional
- location, string, optional
- amenities, string, optional (can also occur more than once if filtering for more than one amenity)
- price_less_than, number, optional
- price_low_to_high, boolean, optional
- price_high_to_low, boolean, optional

Description: View a list of properties based off the search parameters listed above

Methods: GET

Example Responses:

On success:

http://localhost:8000/properties/search/?price_less_than=150&price_low_to_high=True

200OK


```
{
    "count": 2,
    "next": null,
    "previous": null,
    "results": [
        {
            "title": "Property 1",
            "description": "New property",
            "location": "Toronto",
            "num_bed": 1,
            "num_bath": 2,
            "num_guests": 5,
            "price": 100.0
        },
        {
            "title": "Property 3",
            "description": "test",
            "location": "London",
            "num_bed": 2,
            "num_bath": 1,
            "num_guests": 3,
            "price": 150.0
        }
    ]
}
```


Guest Comments

Endpoint: http://localhost:8000/accounts/<int:pk>/comments/

Description: View the comments for guest with id=pk or create a comment for guest with id=pl

Methods: GET, POST

Header (for POST): must add access token as Authorization

Body (for POST):

- content, string, required

Example Responses:

On success for GET:

200OK


```
{
    "count": 2,
    "next": null,
    "previous": null,
    "results": [
        {
            "from_user": 1,
            "content": "test",
            "reply": {
                "from_user": 2,
                "content": "reply",
                "reply": null
            }
        },
        {
            "from_user": 3,
            "content": "new thread",
            "reply": {
                "from_user": 2,
                "content": "replyyyyy",
                "reply": {
                    "from_user": 3,
                    "content": "back again",
                    "reply": null
                }
            }
        }
    ]
}
```


On success for POST

201CREATED


```
{
    "from_user": 3,
    "content": "NEW",
    "reply": null
}
```


POST: if the user is trying to comment on a user that hasn't stayed at their property:

401UNAUTHORIZED

```
[
    "Cannot leave a review on someone that hasn't stayed at your property"
]
```

Guest Comment Reply

Endpoint: http://localhost:8000/accounts/comments/<int:pk>/reply/

Description: Create a reply to the comment with id=pk

Methods: POST

Header: must add access token as Authorization

Body:

- content, string, required

Example Responses:

On success:

201CREATED

```
{
    "from_user": 3,
    "content": "reply",
    "reply": null
}
```

If the user is trying to reply to a comment thread that they are not a part of or the user is trying to reply to their own comment:

401UNAUTHORIZED

```
[
    "Cannot reply to thread that you are not a part of or it is not your turn to comment"
]
```

User is trying to reply to a comment that already has a comment:

400BADREQUEST

```
[
    "This comment already has a reply"
]
```

PropertyComment

Endpoint: http://localhost:8000/properties/<int:pk>/comments/

Description: View comments for property with id=pk or create a comment for property with id=pk

Methods: GET, POST

Header (for POST): must add access token as Authorization

Body (for POST):

- content, string, required

Example Responses:

GET: on success:

200OK


```
{
    "count": 1,
    "next": null,
    "previous": null,
    "results": [
        {
            "from_user": 3,
            "content": "hi",
            "reply": {
                "from_user": 1,
                "content": "hi there",
                "reply": {
                    "from_user": 3,
                    "content": "this is a reply",
                    "reply": null
                }
            }
        }
    ]
}
```


POST: on success:

201CREATED


```
{
    "from_user": 3,
    "content": "hi",
    "reply": null
}
```

POST: If the user doesn't have a reservation for this property or the reservation isn't completed/terminated

401UNAUTHORIZED

```
[
    "Cannot leave a review if you haven't stayed on this property"
]
```

Property Comment Reply

Endpoint: http://localhost:8000/properties/comments/<int:pk>/reply/

Description: Create a reply to the comment with id=pk

Methods: POST

Header: must add access token as Authorization

Body:

- content, string, required

Example Responses:

On success:

201CREATED

```
{
    "from_user": 3,
    "content": "this is a reply",
    "reply": null
}
```

If the user is trying to reply to a comment thread that they are not a part of or the user is trying to reply to their own comment:

401UNAUTHORIZED

```
[
    "Cannot reply to thread that you are not a part of or it is not your turn to comment"
]
```

User is trying to reply to a comment that already has a comment:

400BADREQUEST

```
[
    "This comment already has a reply"
]
```

Properties

CREATE

Endpoint: http://localhost:8000/properties/create/

Description: creates a property

Methods: POST

Header (for POST): must add access token as Authorization (see login)

Body (for POST):

required: title, location, description, location, num_bed, num_bath, num_guests, price

optional: amenities, image

Note: Amenities must be comma separated list of valid amenities

Returns 200 OK on success, 404 Not Found amenity does not exist, or 400 if required field not specified

Example Response

```
{
    "id": 26,
    "data": {
        "title": "new property",
        "description": "blah",
        "location": "blah",
        "num_bed": 1,
        "num_bath": 2,
        "num_guests": 3,
        "price": 999.0
    }
}
```

UPDATE

Endpoint: http://localhost:8000/properties/update/<int:pk>/

Description: updates a property

Methods: PATCH

Header (for PATCH): must add access token as Authorization (see login)

Body (for PATCH):

optional: title, location, description, location, num_bed, num_bath, num_guests, price, amenities, image

Note: Amenities must be comma separated list of valid amenities

Returns 200 OK on success, 404 Not Found amenity does not exist or property with id pk does not exist, 403 if user is not authorized to modify that property (doesn't own it)

Example (updating previous property):

```
{
    "title": "property name updated",
    "description": "blah",
    "location": "blah",
    "num_bed": 1,
    "num_bath": 2,
    "num_guests": 3,
    "price": 999.0
}
```

DELETE

Endpoint: http://localhost:8000/properties/delete/<int:pk>/

Description: deletes a property

Methods: DELETE

Header (for DELETE): must add access token as Authorization (see login)

Returns 200 OK on success, 404 Not Found property with id pk does not exist, 403 if user is not authorized to modify that property (doesn't own it)

Example Response

```
{"property":"Successfully Deleted"}
```

RESERVATIONS

CREATE

Endpoint: http://localhost:8000/reservations/create/<int:pk>/

Description: creates a reservation for property with id pk

Methods: POST

Header (for POST): must add access token as Authorization (see login)

Body (for POST):

required: start_date, end_date

Returns 200 OK on success, 404 Not Found property does not exist

Example Response

```
{
    "id": 4,
    "data": {
        "status": "pending",
        "start_date": "2022-04-21",
        "end_date": "2022-06-09"
    }
}
```

UPDATE

Endpoint: http://localhost:8000/reservations/update/<int:pk>/

Description: updates a reservationwith id pk

Methods: PATCH

Header (for PATCH): must add access token as Authorization (see login)

Body (for PATCH):

required: status

Returns 200 OK on success, 404 Not Found reservation does not exist, 403 Forbidden if user does not own property that reservation references

Example:

```
{
    "status": "approved",
    "start_date": "2022-04-21",
    "end_date": "2022-06-09"
}
```

CANCEL

Endpoint: http://localhost:8000/reservations/terminate/<int:pk>/

Methods: DELETE

Header (for DELETE): must add access token as Authorization (see login)

Returns 204 OK on success, 403 Forbidden if user did not make reservation, 404 Not Found if reservation does not exist

Examples Response:


```
{
    "reservation": "Successfully Deleted"
}
```


LIST

Endpoint: http://localhost:8000/reservations/list/

Methods: GET

Description: Lists all reservation for either user or host

Header (for GET): must add access token as Authorization (see login)

Body:

Optional: property=pk

Note: If query param ‘property’ is not supplied all reservations made by user will be returned. If param is supplied all reservations associated with that property will be returned.

Returns 200 OK on success,

If param supplied: 403 Forbidden if user does not own property, 404 Not Found if property does not exist or state value (if supplied) is not one of ['pending', 'denied', 'expired', 'approved', 'canceled', 'terminated','completed']

Example Response: Listing all users outgoing reservations:

Page 1


```
{
    "count": 18,
    "next": "http://127.0.0.1:8002/properties/reservations/list/?page=2&property=28",
    "previous": null,
    "results": [
        {
            "status": "pending",
            "start_date": "2022-04-20",
```

```
            "end_date": "2022-06-09"
        },
        {
            "status": "pending",
            "start_date": "2022-04-20",
            "end_date": "2022-06-09"
        },
        {
            "status": "pending",
            "start_date": "2022-04-20",
            "end_date": "2022-06-09"
        },
        {
            "status": "pending",
            "start_date": "2022-04-20",
            "end_date": "2022-06-09"
        },
        {
            "status": "pending",
            "start_date": "2022-04-20",
            "end_date": "2022-06-09"
        },
        {
            "status": "pending",
            "start_date": "2022-04-20",
            "end_date": "2022-06-09"
        },
        {
            "status": "pending",
            "start_date": "2022-04-20",
            "end_date": "2022-06-09"
        },
        {
            "status": "pending",
            "start_date": "2022-04-20",
            "end_date": "2022-06-09"
        },
        {
            "status": "pending",
            "start_date": "2022-04-20",
            "end_date": "2022-06-09"
        },
        {
            "status": "approved",
            "start_date": "2022-04-20",
            "end_date": "2022-06-09"
        }
    ]
}
```

```
{
    "count": 18,
    "next": null,
    "previous": "http://127.0.0.1:8002/properties/reservations/list/?property=28",
    "results": [
        {
            "status": "pending",
            "start_date": "2022-04-20",
            "end_date": "2022-06-09"
        },
        {
            "status": "pending",
            "start_date": "2022-04-20",
            "end_date": "2022-06-09"
        },
        {
            "status": "pending",
            "start_date": "2022-04-20",
            "end_date": "2022-06-09"
        },
        {
            "status": "pending",
            "start_date": "2022-04-20",
            "end_date": "2022-06-09"
        },
        {
            "status": "pending",
            "start_date": "2022-04-20",
            "end_date": "2022-06-09"
        },
        {
            "status": "pending",
            "start_date": "2022-04-20",
            "end_date": "2022-06-09"
        },
        {
            "status": "pending",
            "start_date": "2022-04-20",
            "end_date": "2022-06-09"
        },
        {
            "status": "pending",
            "start_date": "2022-04-20",
            "end_date": "2022-06-09"
        }
    ]
}
```

(notice the "approved" reservation is no longer visible now)

Example Response: Listing all reservations made on a property:

```
{
    "count": 3,
    "next": null,
    "previous": null,
    "results": [
        {
            "status": "pending",
            "start_date": "2022-04-20",
            "end_date": "2022-06-09"
        },
        {
            "status": "pending",
            "start_date": "2022-04-20",
            "end_date": "2022-06-09"
        },
        {
            "status": "approved",
            "start_date": "2022-04-20",
            "end_date": "2022-06-09"
        }
    ]
}
```


Then we can filter by state by passing in the query param ‘state=approved’


```
{
    "count": 1,
    "next": null,
    "previous": null,
    "results": [
        {
            "status": "approved",
            "start_date": "2022-04-20",
            "end_date": "2022-06-09"
        }
    ]
}
```