

# Accounts

## Model:

username: Charfield  
password: Charfield  
first\_name: Charfield  
last\_name: Charfield  
phone\_number: Charfield  
biography : Charfield  
guest\_rating: IntegerField  
profile\_picture: ForeignKey  
comments: ForeignKey

## Inherited Fields from other models:

properties: ForeignKey  
guest\_reservations: ForeignKey  
host\_reservations: ForeignKey  
notifications: ForeignKey

**URL:** /accounts/user

**Auth:** Not Authenticated (Post, Get), Authenticated (Put, Delete)

## Post:

Fields: {  
"first\_name": "austin",  
"last\_name": "blackman",  
"email": "austin@mail. Com",  
"username": "austin",  
"password": "password",  
"phone\_number": "555-555-5555"  
}

Description: Create a user within the system

## Put:

All fields except password\_1, password\_2, and profile\_pic will be auto-filled out with current data.

\*if password\_1 is empty, do not update the user's password.

\*if profile\_pic is empty, do not update the user's profile picture

Fields:

{  
"first\_name": "austin",  
"last\_name": "blackman",  
"email": "austin@mail. Com",  
"username": "austin",

```
"password": "password",
"phone_number": "555-555-5555",
"biography": "bio",
"guest_rating" : "0",
"profile_picture" : "",
"password_1": "",
"password_2": ""
}
```

Description: Update a user within the system

#### **Get:**

\* if all is true, ignores pk

Fields:

```
{
"pk": "5",
"all": "false"
}
```

Description: Get a/all user(s) within the system

#### **Delete:**

\*Deletes the currently authenticated user

Fields: None

Description: Delete the currently authenticated user

## **Comments**

#### **Model:**

author: ForeignKey

comment: CharField

rating: IntegerField

content\_type: ForeignKey

object\_id: PositiveIntegerField

content\_object: GenericForeignKey

comments: GenericForeignKey

**URL:** /comments/

**Auth:** Authenticated

#### **Post:**

\* if content\_obj\_type is comment, rating will be ignored

\* content\_obj\_type must be "Account", "Comment", "Property"

Fields:

```
{
"comment": "hi! ",
"rating": "5",
"content_obj_pk": "1",
"content_obj_type": "Account"
}
```

Description: Create a comment as a child on the content\_obj\_type with the given pk

**Put:**

Fields:

```
{
"pk": "1",
"comment": "oops changed my mind! ",
"rating": "5"
}
```

Description: Update the comment with the given pk

**Delete:**

\*Owner must be the deleter

Fields:

```
{
  "pk": "1"
}
```

Description: Delete a comment within the system

**Get:**

\* pk is the comment primary key, if parent\_pk and parent\_type are ignored, gets the single comment. If parent\_pk and parent\_type are present, gets the comments for that object.

parent\_pk is the parent key, parent\_type is "Account", "Property", "Comment"

Fields:

```
{
"pk": "1",
"parent_pk": "",
"parent_type": ""
}
```

Description: Get a single comment, or all comments from a parent

## Images

**Auth:** Authenticated

### Model:

image: ImageField

title: CharField

### Get:

URL: /images/upload

\*MUST BE FORM DATA

Fields:

```
{  
  "title" : "Dog"  
  "image" : image  
}
```

Description: Upload a photo to the system

### Delete:

URL: /images/delete

Fields:

```
{  
  "pk": "1"  
}
```

Description: Delete the image with the given pk

## Properties

### Model

address: CharField

name: Charfield

Owner: ForeignKey

images: ForeignKey

description: ChatField

rating: IntegerField

location: CharField

price\_per\_night: IntegerField

max\_guests: IntegerField

current\_status: CharField choices: "available" or "reserved"

current\_renter: ForeignKey

banned: ManyToManyField: list of banned users by pk/id

bathrooms: IntegerField

bedrooms: IntegerField

backyard: BooleanField  
pool: BooleanField  
wifi: BooleanField  
kitchen: BooleanField  
free\_parking: BooleanField  
pets\_allowed: BooleanField  
Inherited fields from other models: reservations: ManyToManyField

## URL: properties/property

Auth: authenticated

### POST:

Description: create a property within the system

Required fields: {"address", "name", "owner", "images", "description",  
"location", "price\_per\_night", "max\_guests", "bathrooms", "bedrooms"}

optional fields: {"backyard", "pool", "wifi", "kitchen", "free\_parking", "pets\_allowed"}

Note: a property cannot be created with a rating, rating can only be updated after creation.

Example post data:

```
{  
  "address": "308 Negra Arroyo Lane, Albuquerque, New Mexico. 87104",  
  "name": "a very nice place to stay",  
  "owner": "1",  
  "images": "1",  
  "description": "this is a nice place to stay",  
  "location": "Albuquerque",  
  "price_per_night": "200",  
  "max_guests": "4",  
  "bathrooms": "1",  
  "bedrooms": "2"  
}
```

### PUT:

Description: update the fields of an existing property within the system

All fields from the model are optional. Except for pk of the owner. Pk of the property we are trying to edit is required.

Required fields: owner: User -> pk int of that user, pk of the property we are trying to edit

Optional fields

address: str  
name: str

images: Image -> pk int of that image

description: str  
rating: int  
location: str  
price\_per\_night: int  
max\_guests: int

current\_status: str: "available" or "reserved"  
current\_renter: User  
banned: User -> pk int of that user  
bathrooms: int  
bedrooms: int  
backyard: bool  
pool: bool  
wifi: bool  
kitchen: bool  
free\_parking: bool  
pets\_allowed: bool

Example payload:

```
{  
  "pk" : "4",  
  "owner": "1",  
  "description" : "this is the nicest place to stay in canada",  
  "rating": "4",  
  "price_per_night": "350"  
}
```

## Get:

\* if all is true, ignores pk

Fields:

```
{  
  "pk": "5",  
  "all": "false"  
}
```

Description: Get a/all property(ies) within the system

## Delete:

\*Owner must be the deleter

Fields:

```
{  
  "pk" : "1"  
}
```

Description: Delete a Property within the system

## URL: [property/search/](#)

Auth: authenticated

## GET:

## Fields:

"filter\_by": str

"sort\_by": str

"filter\_magnitude": str or int

"sort\_direction": str: ascending or descending"

get the results of a search

for search no fields are required, all are optional.

if no fields are given return paginated results for all properties.

If filter\_by is given, filter magnitude is required.

Example payload:

```
{
    "filter_by": "price",
    "filter_magnitude": "500"
}
```

## Reservations

### Model

```
class State(models.TextChoices):
    PENDING = 'PENDING'
    DENIED = 'DENIED'
    APPROVED = 'APPROVED'
    CANCELED = 'CANCELED'
    TERMINATED = 'TERMINATED'
    COMPLETED = 'COMPLETED'

state: models.CharField (Has to be one of the choices above)
paid: Boolean Value (False by default on initialization)
start_date: DateField
end_date: DateField
guest: ForeignKey to Account Model
host: ForeignKey to Account Model
Property = ForeignKey to Property
```

**URL:** /reservations/

**Auth:** Authenticated (Post, Get), Authenticated (Put)

## Post:

Sample data

```
{
  "guest": "ifaz",
  "property_id": 1,
  "start_date": "10-10-2022",
  "end_date": "10-10-2022"
}
```

Description: Create a reservation request for guest with given username to stay at property with the given property\_id, starting at start\_date and ending at end\_date. User sending the request must be authenticated as the guest account.

## Get:

Get a reservation (or all) from the system

Fields:

```
{
  "reservation_id": "5",
  "all": true,
  "username": "Ifaz",
  "user_type": "guest"
}
```

Description: Returns all reservations belonging to a user with username “ifaz” that have him as a guest. We can symmetrically filter using user\_type: “host”.

For a more specific breakdown of the content that can be passed in the get request:

- If all is true, ignore reservation\_id.
- When all is true, inputting username will make it so that the system gets all reservations belonging to Ifaz, regardless of whether he is a guest or a host. You may choose to narrow down this search to guest or host by using user\_type. You can further narrow this search down by adding a state variable, which has to equal one of the states as defined in the model.
- When all is true and no username is given, all reservations in the system will be shown, particularly useful for superuser or administrative accounts
- Reservation\_id retrieves the reservation with the given id, only viewable if the request user is a host or a guest of that reservation



## PUT:

Update the reservation with the given reservation\_id

The guest for a reservation cannot set the status to APPROVED, nor can they change the paid status of the reservation. Start\_date and end\_date are also optional parameters

```
Mandatory fields: reservation_id
```

```
Optional:
```

- State: check model
- paid: boolean
- start\_date: string in the form MM-DD-YYYY
- end\_date: string in the form MM-DD-YYYY

```
Sample PUT Json data:
```

```
{
    "reservation_id" : "1",
    "state" : "approved",
    "paid" : true
}
```

## Delete:

Deletes the notification with the given reservation\_id from the system

Payload:

```
{
    "reservation_id": "1"
}
```

## Notifications Model

```
class Notification(models.Model):
    """A notification sent to a user"""
    SEEN_CHOICES = [
        (0, 'Unseen'),
        (1, 'Seen'),
    ]
    """
    Notification_type must be one of the following values:
```

```

        - "reservation_request"
        - "cancellation_request"
        - "reservation_approved"
        - "cancellation_approved"
        - "property_comment"
        - "approval_request"
        - "cancellation_request"
    """

NOTIFICATION_TYPE_CHOICES = [
    ('reservation_request', 'You have a Reservation Request'),
    ('cancellation_request', 'You have a Cancellation Request'),
    ('reservation_approved', 'Your Reservation Request was Approved'),
    ('cancellation_approved', 'Your Cancellation Request Approved'),
    ('property_comment', 'Someone left a comment on your property!'),
    ('test_notification', 'Test notification from Phase 2 of Restify!'),
]

# link the notification to the associated account
recipient = models.ForeignKey(Account, on_delete=models.CASCADE,
related_name='notifications')

# type of the notification
notification_type = models.CharField(choices=NOTIFICATION_TYPE_CHOICES,
max_length=50, default='test_notification')

# track whether the notification has been seen or not (0 or 1)
# 0 by default
seen = models.IntegerField(choices=SEEN_CHOICES, default=0)

# a link associated with the notification
link = models.URLField(default="http://google.com/")

```

URL: /notifications/

Authentication: POST, GET, PUT, DELETE

POST:

"""

Send a notification to a user with the specified username.

Required Fields: username

(notification\_type and link are optional and have default values)

Ideally Notification\_type must be one of the following values:

- "reservation\_request"
- "cancellation\_request"
- "reservation\_approved"

- "cancellation\_approved"
  - "property\_comment"
  - "approval\_request"
  - "cancellation\_request"
- By default it is "test\_notification"

The "link" field is optional and defaults to "https://www.google.com" if not provided.

Notification\_type is optional too and defaults to 'test\_notification' if not given. A user cannot send a comment to themselves.

Payload format (JSON) POST:

```
{
  "username": "ifaz",
  "notification_type": "new_reservation",
  "link": "https://www.restify.com",
}
```

""

GET:

""

Get a notification from the system.

Payload variables and their purposes:

All is a required field. is is treated as a switch variable

- all: True or False - whether or not you want to display all notifications in the system
- username: the username of the account whose notifications we wish to filter.
- notification\_id: the id of the notification we want to look up

Cases:

When all is true notification\_id is ignored

All is true and username is valid: Retrieve all notifications belonging to the account with the specified username

All is true and username is not given: Retrieve all notifications from the system

All is true and username is invalid: Give an appropriate error

When all is false notification\_id is required.

Payload variations:

Get all notifications belonging to ifaz

```
{
  "username": "ifaz",
```

```
"all" : true
}
```

Get all notifications in the system

```
{
  "all": true,
}
```

Get one notification from the system if it exists

```
{
  "notification_id": 1
}
```

```
""
```

## PUT

```
""
```

Edit the notification with the given notification\_id (Purpose is to read / unread a notification, setting seen to 0 or 1)

Mandatory field: notification\_id, seen

Payload format (JSON) PUT:

```
{
  "notification_id": 1,
  "seen": 1
}
""
```

## DELETE

```
""
```

Delete the notification with the given notification\_id

Mandatory field: notification\_id

DELETE request format

```
{
  "notification_id": "1"
}
""
```