Authentication - Token

Get token (with login credentials)

URL: `POST api/token/obtain/`

Request body:

Param	Type	Description
username	String	
password	String	

Sample Request:

```
{
  "username": "jeffbezos",
  "password": "p@ssw0rd"
}
```

Response body:

Param	Type	Description
refresh	String	
access	String	
id	Integer	User ID

Sample Response:

```
{
  "refresh": "eyJ0eXAiOiJKV1Q …",
  "access": "eyJ0eXAiOiJKV1Q …"
  "id": 5
}
```

Get token (with Google ID token)

URL: `POST api/token/obtain/social/`

Request body:

Param	Type	Description
idToken	String	

Sample Request:

```
{"idToken": "loremipsum123"}
```

Response body:

Param	Type	Description
refresh	String	
access	String	
userId	Integer	
isNew	Boolean	

Sample Response:

```
{
  "refresh": "eyJ0eXAiOiJKV1Q …",
  "access": "eyJ0eXAiOiJKV1Q …"
  "id": 5,
  "isNew": True
}
```

Get token (with refresh token)

URL: `POST api/token/refresh/`

Param	Type	Description
refresh	String	

Sample Request:

```
{"refresh": "eyJ0eXAiOiJKV1Q …"}
```

Response body:

Param	Type	Description
refresh	String	
access	String	

Sample Response:

```
{
  "refresh": "eyJ0eXAiOiJKV1Q …",
  "access": "eyJ0eXAiOiJKV1Q …"
}
```

Authentication - User

List/ Filter all users

URL: `GET /api/user/search?keywords={searchKeywords}&offset={offset}&limit={limit}`

Description: Returns users, filtered by keywords. username, firstName or lastName must start with keyword.

Request body: None

Response body: Array of User.
Refer to Get a user for format of User.

Get a user

URL: `GET api/user/{username}/`

Description: Returns user matching the provided username

Request body: None

Response body: User

Param	Type	Description
userId	Integer	
email	String	
username	String	
firstName	String	
lastName	String	
pictureUrl	String	
description	String	
likesCount	Integer	Total number of likes user has accumulated across all items
collectionsCount	Integer	Number of collections user possesses
itemsCount	Integer	Total number of items user possesses across all collections

Sample Response:

```
{
  "userId": 12,
  "email": “asdasd@sad.com”,
  "username": "PokemonMaster69",
  "firstName": "Master",
  "lastName": "Baiter",
  "pictureUrl": "example.com/image.jpg",
  "description": "I am a pokemon master",
  "likesCount": 14,
  "collectionsCount": 2,
  "itemsCount": 10,
}
```

Get current user

URL: `GET api/user/`

Description: Get the profile of the authorization token owner

Request body: None

Response body: User.
Refer to Get a user for format of User.

Create user

URL: `POST /api/user/create/`

Description: Create a new user in the database.

Request body:

Param	Type	Description
email	String	
username	String	
password	String	

Sample Request:

```
{
  "email": "test@example.com",
  "username": "jeffbezos",
  "password": "p@ssw0rd"
}
```

Response body:

Param	Type	Description
id	Integer	
email	String	
username	String	

Sample response:

```
{
  "id": 123
  "email": "test@example.com",
  "username": "jeffbezos”
}
```

Delete user

URL: `POST /api/user/create/`

Description: Delete a new user from the database.

Request body: None (authentication token used)

Response body: None

Categories

List all categories

URL: `GET /categories?indicateEmpty={true/false}`

Description: Returns a list of all categories, sorted alphabetically.
If `indicateEmpty=true`, there is an additional `isEmpty` return param to indicate if there are no items belonging to this category. Furthermore, the list of categories will be sorted with non-empty categories appearing first.

Request body: None

Response body: Array of Category.
See sample response for example of Category.

Sample response body:

```
[
  {
    "categoryId": 19,
    "name": "Pokemon Cards"
    "isEmpty": false
  },
  {
    "categoryId": 29,
    "name": "Sneakers"
    "isEmpty": false
  }
]
```

Chat

Get chat token

URL: `POST chat/`

Request body: None (authentication token used)

Response body:

Param	Type	Description
chatId	String	collectify user ID
chatToken	String	Stream Chat token

Collections

List/ Filter all collections

URL: `GET /collections?keywords={searchKeyword}&category={categoryId}`
`&user={userId}&followed={true/false}`
`&offset={offset}&limit={limit}`

Description: Returns collections (with coverImages), filtered by any of:

- Keyword (search by collection name, collection description, category name)
- Category
- User
- Is followed

If search keywords are provided, collections are sorted by search rank. Else, collections are sorted by the latest created collection first.

Request body: None

Response body: Array of Collection (with coverImages).
A Collection (with coverImages) is similar to a Collection, but includes the coverImage of the latest 3 items in the collection. coverImage is the thumbnail of the earliest uploaded image in the item.
See sample response for example of Collection (with coverImages).
Refer to Get a collection for format of Collection.

Sample response body:

```
[
  {
    "collectionId": 123,
    "collectionName": "1st Gen Pokemon Cards",
    "collectionDescription": "My collection of 1st gen cards",
    "collectionCreationDate":"2021-09-23T01:22:47.541Z",
    "ownerId": 12,
    "ownerUsername": "pokemaster69"
    "categoryId": 19,
    "categoryName": "Pokemon Cards",
    "isFollowed": true,
    "followersCount": 10,
    "coverImages":[
      "https://s3.amazonaws.com/bucket/image1.jpg",
      "https://s3.amazonaws.com/bucket/image2.jpg",
      "https://s3.amazonaws.com/bucket/image5.jpg"
    ]
  }
]
```

Get a collection

URL: `GET /collections/{collectionId}/`

Description: Returns a collection

Request body: None

Response body: Collection

Param	Type	Description
collectionId	Integer	
collectionName	String	
collectionDescription	String	
collectionCreationDate	String	JavaScript's Date.toJSON() format
ownerId	Integer	
ownerUsername	String	
categoryId	Integer	
categoryName	String	
isFollowed	Boolean	
followersCount	Integer	

Sample Response:

```
{
  "collectionId": 123,
  "collectionName": "1st Gen Pokemon Cards",
  "collectionDescription": "My collection of 1st gen cards",
  "collectionCreationDate": "2021-09-23T01:22:47.541Z",
  "ownerId": 12,
  "ownerUsername": "pokemaster69"
  "categoryId": 19,
  "categoryName": "Pokemon Cards",
  "isFollowed": true,
  "followersCount": 10,
}
```

Create new collection

URL: `POST /collections/`

Description: Creates a new collection

Request body:

Param	Type	Description
collectionName	String	
collectionDescription	String	Optional
categoryId	Integer	

Response body: Newly created Collection.
Refer to Get a collection for format of Collection.

Update collection

URL: `PATCH /collections/{collectionId}/`

Description: Update existing collection

Request body:

Param	Type	Description
collectionName	String	
collectionDescription	String	
categoryId	Integer	

Response body: None

Delete collection

URL: `DELETE /collections/{collectionId}/`

Description: Deletes existing collection, along with all items in collection

Request body: None

Response body: None

Followers

List followers of a collection

URL: `GET /followers?collection={collectionId}&offset={offset}&limit={limit}`

Request body: None

Response body: Array of Follower.
Refer to sample response for example of Follower.

Sample Response:

```
[
  {
    "username": "jeffbezos",
    "userId": 5,
    "profilePictureUrl": "example.com/image.jpg"
  },
  {
    "username": "pokemaster69",
    "userId": 123,
    "profilePictureUrl": "example.com/image.jpg"
  },
]
```

Follow a collection

URL: `POST /followers/`

Request body:

Param	Type	Description
collectionId	Integer	

Response body: Newly added follower.
Refer to List followers of a collection for example of Follower.

Unfollow a collection

URL: `DELETE /followers?collection={collectionId}`

Request body: None

Response body: None

Items

List/ Filter all items

URL: `GET /items?keywords={searchKeywords}&category={categoryId}`
`&liked={true/false}&followed={true/false}`
`&detailed={true/false}&discover={true/false}`
`&isTradable={true/false}&offset={offset}&limit={limit}`

Description: Returns items, filtered by any of:

- Keyword (search by item name, item description, collection name, collection description, category name)
- Category
- Is followed
- Is liked
- Is tradable

If `detailed=true` then detailed view is returned, else simple view is returned.
If `discover=true` then the user's own items are excluded.
If search keywords are provided, items are sorted by search rank. Else, items are sorted by the latest created item first.

Request body: None

Response body: Array of Item (simple view) OR Array of Item (detailed view)
Refer to List items in a collection for format of Item (simple view).
Refer to Get an item for format of Item (detailed view).

List items in a collection

URL: `GET /collections/{collectionId}/items`
`?offset={offset}&limit={limit}`

Description: Returns items (simple view) belonging to a collection.
Items are sorted by the latest created item first.

Request body: None

Response body: Array of Item (simple view).
See sample response for example of Item (simple view).
coverImage is the thumbnail of the earliest uploaded image in the item.

Sample Response body:


```
[
  {
    "itemId": 500,
    "itemName": "Shiny Charmander",
    "itemDescription": "My first ever card!",
```

```
    "itemCreationDate": "2021-09-23T01:22:47.541Z",
    "ownerId": 28,
    "ownerUsername":"pokemaster69",
    "collectionId": 123,
    "collectionName":"Cardz!,
    "isTradable": false
    "isPopular": true
    "coverImage": "https://s3.amazonaws.com/bucket/image1.jpg"
  },
]
```

Get an item

URL: `GET /collections/{collectionId}/items/{itemId}/`

Description: Returns an item.
Images in an item are sorted by the earliest uploaded image first.

Request body: None

Response body: Item (detailed view)

Param	Type	Description
itemId	Integer	
itemName	String	
itemDescription	String	
itemCreationDate	String	JavaScript Date.toJSON() format
ownerId	Integer	
ownerUsername	String	
collectionId	Integer	
collectionName	String	
isTradable	Boolean	
isPopular	Boolean	Items with more than twice the average number of likes are considered popular
isLiked	Boolean	
likesCount	String	
images	Array of Image	See sample response for example.

Sample Response:

```
{
  "itemId": 501,
  "itemName": "Rare Mew Two",
  "itemDescription": "Bought for $100",
  "itemCreationDate": "2021-09-23T01:22:47.541Z",
  "ownerId": 28,
  "ownerUsername":"pokemaster69",
  "collectionName":"Cardz!,
  "isTradable": false
  "isPopular": true
  "isLiked": true,
  "likesCount": 4
  "images": [
    {
      "imageId": 2,
      "imageUrl": "https://s3.amazonaws.com/bucket/image2.jpg",
      "imageUploadDate": "2021-09-23T01:22:47.541Z"
    },
    {
      "imageId": 3,
      "imageUrl": "https://s3.amazonaws.com/bucket/image3.jpg",
      "imageUploadDate": "2021-09-23T01:22:47.541Z"
    }
  ]
}
```

Create new item

URL: `POST /collections/{collectionId}/items/`

Description: Create a new item

Request body:

Param	Type	Description
itemName	String	
itemDescription	String	Optional
isTradable	Boolean	
images	Array of Files	Image files belonging to item, in order of display

Response body: Newly created Item. Refer to Get an item for format of Item.
Note that param “images” is empty only in this initial response.

Update item

URL: `PATCH /collections/{collectionId}/items/{itemId}/`

Description: Returns an item

Request body:

Param	Type	Description
itemName	String	
itemDescription	String	
isTradable	Boolean	
updatedCollection	Integer	Modify parent collection
newImages	Array of Files	New images are added to the back of item
deletedImageIds	Array of Int	

Response body: None

Delete item

URL: `DELETE /collections/{collectionId}/items/{itemId}/`

Description: Deletes existing item, along with all images in collection (database records of image deleted, but image still exists in S3).

Request body: None

Response body: None

Likes

List likes of an item

URL: `GET /likes?item={itemId}&offset={offset}&limit={limit}`

Request body: None

Response body: Array of Like.
Refer to sample response for example of Like.

Sample Response:

```
[
  {
    "username": "jeffbezos",
    "userId": 5,
    "profilePictureUrl": "example.com/image.jpg"
  },
  {
    "username": "pokemaster69",
    "userId": 123,
    "profilePictureUrl": "example.com/image.jpg"
  },
]
```

Like a collection

URL: `POST /likes/`

Request body:

Param	Type	Description
itemId	Integer	

Response body: Like.
Refer to List likes of an item for example of Like.

Unlike a collection

URL: `DELETE /likes?item={itemId}`

Request body: None

Response body: None