Shopping Deals Tracker API Documentation

Overview

The aim of the Shopping Deals Tracker project is to create a platform where users can track and discover various deals and discounts offered to customers. The application will provide users with a seamless experience to browse, search, save and claim deals based on their preferences and interests.

Base URL

The base URL for the API is **https://api.shoppingdeals.com/v1**.
All API requests should be made to this base URL, followed by the specific endpoint you need to access.

Request Headers

The following headers should be included in the API requests:

Content-Type

This header specifies the media type of the request body. It is required for POST and PUT requests when sending data in the request body.

Example:

```
Content-Type: application/json
```

Accept

This header specifies the media type of the expected response. It is recommended to set this header to ensure the API returns the desired response format.

Example:

```
Accept: application/json
```

Account Management

Introduction

The Shopping Deals Tracker API is a RESTful API that allows users to manage their account information. The API provides endpoints for creating, retrieving, updating, and deleting user accounts. This documentation will guide you through the various endpoints and their usage, so you can seamlessly integrate the API into your application.

Shopping Deals Tracker API Documentation

Endpoints

1. Get All Accounts

This endpoint allows you to retrieve a list of all user accounts in the system.

Endpoint

```
GET /accounts
```

Request Parameters This endpoint does not accept any request parameters.

Response

```
{
    "data": [
        {
            "id": "6061c1b9b3d1e40015c4a1a1",
            "username": "johndoe",
            "email": "johndoe@example.com",
            "createdAt": "2023-04-07T00:00:00.000Z"
        },
        {
            "id": "6061c1b9b3d1e40015c4a1a2",
            "username": "janedoe",
            "email": "janedoe@example.com",
            "createdAt": "2023-04-05T00:00:00.000Z"
        }
    ],
    "status": "success"
}
```

The data field in the response contains an array of user account objects, each with the following properties:

- id: a unique identifier for the user account
- username: the username associated with the user account
- email: the email address associated with the user account
- createdAt: the timestamp when the user account was created

You can use this endpoint to retrieve a list of all registered users in the system, which can be helpful for administrative tasks or reporting purposes.

Shopping Deals Tracker API Documentation

2. Create an Account

This endpoint allows you to create a new user account.

Endpoint

```
POST /accounts
```

Request Body

```
{
    "username": "johndoe",
    "password": "password123",
    "email": "johndoe@example.com"
}
```

The request body should contain the following properties:

- username: the desired username for the new user account
- password: the password for the new user account
- email: the email address for the new user account

Response

```
{
    "data": {
        "id": "6061c1b9b3d1e40015c4a1a3",
        "username": "johndoe",
        "email": "johndoe@example.com",
        "createdAt": "2023-04-07T12:00:00.000Z"
    },
    "status": "success"
}
```

The response includes the newly created user account, with the following properties:

- id: a unique identifier for the user account
- username: the username associated with the user account
- email: the email address associated with the user account
- createdAt: the timestamp when the user account was created

You can use this endpoint to allow users to sign up for your application and create new accounts.

Shopping Deals Tracker API Documentation

3. Get Account by ID
This endpoint allows you to retrieve the details of a specific user account by its ID.

Endpoint

```
GET /accounts/{id}
```

Request Parameters

- id (required): the unique identifier of the user account you want to retrieve

Response

```
{
    "data": {
        "id": "6061c1b9b3d1e40015c4a1a1",
        "username": "johndoe",
        "email": "johndoe@example.com",
        "createdAt": "2023-04-07T00:00:00.000Z"
    },
    "status": "success"
}
```

The response includes the details of the requested user account, with the following properties:

- id: a unique identifier for the user account
- username: the username associated with the user account
- email: the email address associated with the user account
- createdAt: the timestamp when the user account was created

You can use this endpoint to retrieve the details of a specific user, for example, when displaying the user's account information in your application.

4. Update an Account
This endpoint allows you to update an existing user account.

Endpoint

```
PUT /accounts/{id}
```

Request Parameters

- id (required): the unique identifier of the user account you want to update

Request Body

```
{
    "username": "johndoe_updated",
    "email": "johndoe_updated@example.com"
}
```

The request body can contain the following properties, which will be used to update the user account:

- username: the updated username for the user account
- email: the updated email address for the user account

Response

```
{
    "data": {
        "id": "6061c1b9b3d1e40015c4a1a1",
        "username": "johndoe_updated",
        "email": "johndoe_updated@example.com",
        "createdAt": "2023-04-07T00:00:00.000Z"
    },
    "status": "success"
}
```

The response includes the updated user account information, with the username and email fields reflecting the changes made.

You can use this endpoint to allow users to update their account information, such as their username or email address.

5. Delete an Account
This endpoint allows you to delete a user account.

Endpoint

```
DELETE /accounts/{id}
```

Request Parameters

- id (required): the unique identifier of the user account you want to delete

Response

```json
{
    "data": {
        "id": "6061c1b9b3d1e40015c4a1a1",
        "username": "johndoe_updated",
        "email": "johndoe_updated@example.com",
        "createdAt": "2023-04-07T00:00:00.000Z"
    },
    "status": "success"
}
```

The response includes the details of the deleted user account.

You can use this endpoint to allow users to delete their own accounts, or to allow administrators to remove user accounts from the system.

Error Handling

The Shopping Deals Tracker API uses standard HTTP status codes to indicate the success or failure of a request. If a request fails, the API will return a JSON response with an error field that contains a description of the error.

Example error response:

```json
{
    "error": "Account not found"
}
```

You can use this error information to provide meaningful feedback to your users or handle the error appropriately in your application.

Deal Management

Introduction

This RESTful API allows our Admin and UI teams to seamlessly manage deals and discounts for our Shopping Deals Tracker application. Whether you need to retrieve a list of all available deals, create a new deal, update an existing one, or even delete a deal.

Shopping Deals Tracker API Documentation

Endpoints

1. Get All Deals

This endpoint allows you to retrieve a list of all deals in the system.

Endpoint

```
GET /deals
```

Response

```
{
    "data": [
        {
            "id": "6061c1b9b3d1e40015c4a1a1",
            "name": "50% off on Nike Shoes",
            "price": 49.99,
            "category": "Clothing",
            "shortDesc": "Get 50% off on selected Nike
shoes",
            "longDesc": "Don't miss out on this amazing
deal! Get 50% off on all Nike shoes in our store.",
            "couponCode": "NIKE50",
            "createdAt": "2023-04-07T00:00:00.000Z",
            "expiryDate": "2023-06-30T00:00:00.000Z",
            "image": "https://example.com/nike-shoes.jpg"
        },
        {
            "id": "6061c1b9b3d1e40015c4a1a2",
            "name": "Buy one, get one free on Levi's
Jeans",
            "price": 59.99,
            "category": "Clothing",
            "shortDesc": "Buy one Levi's jean, get another
one for free",
            "longDesc": "Treat yourself or a loved one with
this amazing deal on Levi's jeans. Buy one, get one free!",
            "couponCode": "LEVIS-BOGO",
```

Shopping Deals Tracker API Documentation

```
                "createdAt": "2023-04-05T00:00:00.000Z",
                "expiryDate": "2023-05-31T00:00:00.000Z",
                "image": "https://example.com/levis-jeans.jpg"
            }
        ],
    "status": "success"
}
```

The data field in the response contains an array of deal objects, each with the following properties:

- id: a unique identifier for the deal
- name: the name of the deal
- price: the discounted price of the deal
- category: the category of the deal (e.g., Clothing, Electronics, Home)
- shortDesc: a brief description of the deal
- longDesc: a more detailed description of the deal
- couponCode: the coupon code associated with the deal
- createdAt: the timestamp when the deal was created
- expiryDate: the date when the deal expires
- image: the URL of the deal's image

You can use this endpoint to display a list of all available deals on the Shopping Deals Tracker application.

2. Create a Deal

To create a new deal, you'll need to send a POST request to the /deals endpoint with the necessary deal information in the request body.

Endpoint

```
POST /deals
```

Request Body

```
{
    "name": "50% off on Nike Shoes",
    "price": 49.99,
    "category": "Clothing",
    "shortDesc": "Get 50% off on selected Nike shoes",
    "longDesc": "Don't miss out on this amazing deal! Get 
50% off on all Nike shoes in our store.",
    "couponCode": "NIKE50",
    "expiryDate": "2023-06-30T00:00:00.000Z",
    "image": "https://example.com/nike-shoes.jpg"
}
```

Response

```
{
    "data": {
        "id": "6061c1b9b3d1e40015c4a1a3",
        "name": "50% off on Nike Shoes",
        "price": 49.99,
        "category": "Clothing",
        "shortDesc": "Get 50% off on selected Nike shoes",
        "longDesc": "Don't miss out on this amazing deal!
Get 50% off on all Nike shoes in our store.",
        "couponCode": "NIKE50",
        "createdAt": "2023-04-07T12:00:00.000Z",
        "expiryDate": "2023-06-30T00:00:00.000Z",
        "image": "https://example.com/nike-shoes.jpg"
    },
    "status": "success"
}
```

The response includes the newly created deal, with its unique id, name, price, category, shortDesc, longDesc, couponCode, createdAt, expiryDate, and image properties. This data will be essential for our UI team, as they can use it to display the deal information and allow users to take advantage of the discounts.

3. Get Deal by ID

If you need to retrieve the details of a specific deal, you can use the GET /deals/{id} endpoint, where {id} is the unique identifier of the deal you're looking for.

Endpoint

```
GET /deals/{id}
```

Response

```json
{
    "data": {
        "id": "6061c1b9b3d1e40015c4a1a1",
        "name": "50% off on Nike Shoes",
        "price": 49.99,
        "category": "Clothing",
        "shortDesc": "Get 50% off on selected Nike shoes",
        "longDesc": "Don't miss out on this amazing deal! Get 50% off on all Nike shoes in our store.",
        "couponCode": "NIKE50",
        "createdAt": "2023-04-07T00:00:00.000Z",
        "expiryDate": "2023-06-30T00:00:00.000Z",
        "image": "https://example.com/nike-shoes.jpg"
    },
    "status": "success"
}
```

The response includes the details of the requested deal, including the id, name, price, category, shortDesc, longDesc, couponCode, createdAt, expiryDate, and image properties. This endpoint will be particularly useful for our UI team, as they can use it to display the deal's details in the application.

4. Update a Deal

To update an existing deal, you'll need to send a PUT request to the /deals/{id} endpoint, where {id} is the unique identifier of the deal you want to update. In the request body, include the updated deal information.

Endpoint

```
PUT /deals/{id}
```

Request Body

```json
{
    "name": "60% off on Nike Shoes",
    "price": 39.99,
    "expiryDate": "2023-07-31T00:00:00.000Z"
}
```

Response

```
{
    "data": {
        "id": "6061c1b9b3d1e40015c4a1a1",
        "name": "60% off on Nike Shoes",
        "price": 39.99,
        "category": "Clothing",
        "shortDesc": "Get 60% off on selected Nike shoes",
        "longDesc": "Don't miss out on this amazing deal!
Get 60% off on all Nike shoes in our store.",
        "couponCode": "NIKE60",
        "createdAt": "2023-04-07T00:00:00.000Z",
        "expiryDate": "2023-07-31T00:00:00.000Z",
        "image": "https://example.com/nike-shoes.jpg"
    },
    "status": "success"
}
```

The response includes the updated deal information, with the name, price, and expiryDate fields reflecting the changes made. This endpoint will be useful for our Admin team, who may need to update deal information as part of their deal management duties.

5. Delete a Deal

If you need to delete a deal, you can use the DELETE /deals/{id} endpoint, where {id} is the unique identifier of the deal you want to delete.

Endpoint

```
DELETE /deals/{id}
```

Response

```
{
    "data": {
        "id": "6061c1b9b3d1e40015c4a1a1",
        "name": "60% off on Nike Shoes",
        "price": 39.99,
        "category": "Clothing",
        "shortDesc": "Get 60% off on selected Nike shoes",
        "longDesc": "Don't miss out on this amazing deal!
Get 60% off on all Nike shoes in our store.",
        "couponCode": "NIKE60",
        "createdAt": "2023-04-07T00:00:00.000Z",
        "expiryDate": "2023-07-31T00:00:00.000Z",
        "image": "https://example.com/nike-shoes.jpg"
    },
    "status": "success"
}
```

The response includes the details of the deleted deal, which can be useful for our Admin team's audit and reporting purposes.

6. Filter Deals by Price
This endpoint allows you to retrieve deals based on a specified price range.

Endpoint

```
GET /deals/filtered/price
```

Request Body

```
{
    "min": 30,
    "max": 50
}
```

Response

```
{
    "data": [
        {
            "id": "6061c1b9b3d1e40015c4a1a1",
            "name": "60% off on Nike Shoes",
            "price": 39.99,
            "category": "Clothing",
            "shortDesc": "Get 60% off on selected Nike
shoes",
            "longDesc": "Don't miss out on this amazing
deal! Get 60% off on all Nike shoes in our store.",
            "couponCode": "NIKE60",
            "createdAt": "2023-04-07T00:00:00.000Z",
            "expiryDate": "2023-07-31T00:00:00.000Z",
            "image": "https://example.com/nike-shoes.jpg"
        }
    ],
    "status": "success"
}
```

The data field in the response contains an array of deal objects that match the specified price range. This endpoint will be useful for our UI team, as they can use it to allow users to filter deals based on their budget.

7. Filter Deals by Category
This endpoint allows you to retrieve deals based on a specified category.
Endpoint

```
GET /deals/filtered/tags
```

Request Body

```
{
    "tags": "Clothing Electronics"
}
```

Response

```
{
    "data": [
        {
            "id": "6061c1b9b3d1e40015c4a1a1",
            "name": "60% off on Nike Shoes",
            "price": 39.99,
            "category": "Clothing",
            "shortDesc": "Get 60% off on selected Nike
shoes",
            "longDesc": "Don't miss out on this amazing
deal! Get 60% off on all Nike shoes in our store.",
            "couponCode": "NIKE60",
            "createdAt": "2023-04-07T00:00:00.000Z",
            "expiryDate": "2023-07-31T00:00:00.000Z",
            "image": "https://example.com/nike-shoes.jpg"
        },
        {
            "id": "6061c1b9b3d1e40015c4a1a4",
            "name": "20% off on Samsung TV",
            "price": 799.99,
            "category": "Electronics",
            "shortDesc": "Get 20% off on selected Samsung
TVs",
            "longDesc": "Don't miss out on this amazing
deal! Get 20% off on all Samsung TVs in our store.",
            "couponCode": "SAMSUNG20",
            "createdAt": "2023-04-01T00:00:00.000Z",
            "expiryDate": "2023-06-15T00:00:00.000Z",
            "image": "https://example.com/samsung-tv.jpg"
        }
    ],
    "status": "success"
}
```

The data field in the response contains an array of deal objects that match the specified categories. This endpoint will be useful for our UI team, as they can use it to allow users to filter deals based on their interests.

Error Handling

The Shopping Deals Tracker API uses standard HTTP status codes to indicate the success or failure of a request. If a request fails, the API will return a JSON response with an error field that contains a description of the error.

Example error response:

```
{
    "error": "Deal not found"
}
```

You can use this error information to provide meaningful feedback to your users or handle the error appropriately in your application.