# Documentation for `getAllPrompts` API Endpoint

The `getAllPrompts` API endpoint is a part of the `swarms.world` application, designed to fetch all prompt records from the database. This endpoint is crucial for retrieving various prompts stored in the `swarms_cloud_prompts` table, including their metadata such as name, description, use cases, and tags.

## Purpose

The primary purpose of this API endpoint is to provide a method for clients to fetch a list of prompts stored in the `swarms_cloud_prompts` table, with the ability to filter by name, tags, and use cases.

## API Endpoint Definition

### Fetch All Prompts

#### Endpoint URL

```
https://swarms.world/get-prompts
```

#### HTTP Method

```
GET
```

~~~

Query Parameters

- **name** (optional): A substring to match against the prompt name. The query is case-insensitive.
- **tag** (optional): A comma-separated list of tags to filter prompts by. The query matches any of the provided tags, and is case-insensitive.
- **use_case** (optional): A substring to match against the use case titles within the `use_cases` array. The query is case-insensitive.
- **use_case_description** (optional): A substring to match against the use case descriptions within the `use_cases` array. The query is case-insensitive.

Response

Success Response (200)

Returns an array of prompts.

```json
[
  {
    "id": "string",
    "name": "string",
    "description": "string",
    "prompt": "string",
    "use_cases": [
~~~

      {
        "title": "string",
        "description": "string"
      }
    ],
    "tags": "string"
  },
  ...
]

##### Error Responses

- **405 Method Not Allowed**

  ```json
  {
    "error": "Method <method> Not Allowed"
  }
  ```

- **500 Internal Server Error**

  ```json
  {
    "error": "Could not fetch prompts"
  ```

  }
  ```

Fetch Prompt by ID

Endpoint URL

```
https://swarms.world/get-prompts/[id]
```

#### HTTP Method

```
GET
~~~

#### Response

##### Success Response (200)

Returns a single prompt by ID.

```json
{
  "id": "string",
~~~

```
  "name": "string",
  "description": "string",
  "prompt": "string",
  "use_cases": [
    {
      "title": "string",
      "description": "string"
    }
  ],
  "tags": "string"
}
```

##### Error Responses

- **404 Not Found**

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

- **500 Internal Server Error**

  ```json
  ```

  {
    "error": "Could not fetch prompt"
  }
  ```

### Request Handling

1. **Method Validation**: The endpoint only supports the `GET` method. If a different HTTP method is used, it responds with a `405 Method Not Allowed` status.

2. **Database Query**:

   - **Fetching All Prompts**: The endpoint uses the `supabaseAdmin` client to query the `swarms_cloud_prompts` table. Filters are applied based on the query parameters (`name`, `tag`, and `use_cases`).
   - **Fetching a Prompt by ID**: The endpoint retrieves a single prompt from the `swarms_cloud_prompts` table by its unique ID.

3. **Response**: On success, it returns the prompt data in JSON format. In case of an error during the database query, a `500 Internal Server Error` status is returned. For fetching by ID, if the prompt is not found, it returns a `404 Not Found` status.

### Code Example

#### JavaScript (Node.js)
  ```

```javascript
import fetch from "node-fetch";

// Fetch all prompts with optional filters
const getPrompts = async (filters) => {
  const queryString = new URLSearchParams(filters).toString();
  const response = await fetch(
    `https://swarms.world/get-prompts?${queryString}`,
    {
      method: "GET",
    }
  );

  if (!response.ok) {
    throw new Error(`Error: ${response.statusText}`);
  }

  const data = await response.json();
  console.log(data);
};

// Fetch prompt by ID
const getPromptById = async (id) => {
  const response = await fetch(`https://swarms.world/get-prompts/${id}`, {
    method: "GET",
  });
  if (!response.ok) {
    throw new Error(`Error: ${response.statusText}`);
  }

  const data = await response.json();
  console.log(data);
};

// Example usage
getPrompts({
  name: "example",
  tag: "tag1,tag2",
  use_case: "example",
  use_case_description: "description",
}).catch(console.error);
getPromptById("123").catch(console.error);
```

#### Python

```python
import requests

# Fetch all prompts with optional filters
def get_prompts(filters):
    response = requests.get('https://swarms.world/get-prompts', params=filters)

    if response.status_code != 200:
        raise Exception(f'Error: {response.status_code}, {response.text}')

    data = response.json()
    print(data)

# Fetch prompt by ID
def get_prompt_by_id(id):
    response = requests.get(f'https://swarms.world/get-prompts/{id}')

    if response.status_code != 200:
        raise Exception(f'Error: {response.status_code}, {response.text}')

    data = response.json()
    print(data)

# Example usage
get_prompts({'name': 'example', 'tag': 'tag1,tag2', 'use_case': 'example', 'use_case_description': 'description'})
get_prompt_by_id('123')
```

#### cURL

```sh
# Fetch all prompts with optional filters
curl -X GET
"https://swarms.world/get-prompts?name=example&tag=tag1,tag2&use_case=example&use_case_description=description"

# Fetch prompt by ID
curl -X GET https://swarms.world/get-prompts/123
```

Go

```go
package main

import (
    "fmt"
    "io/ioutil"
    "net/http"
    "net/url"
)

func getPrompts(filters map[string]string) {
    baseURL := "https://swarms.world/get-prompts"
    query := url.Values{}
    for key, value := range filters {
        query.Set(key, value)
    }
    fullURL := fmt.Sprintf("%s?%s", baseURL, query.Encode())

    resp, err := http.Get(fullURL)
    if err != nil {
        panic(err)
    }
    defer resp.Body.Close()

    if resp.StatusCode != http.StatusOK {
        body, _ := ioutil.ReadAll(resp.Body)
        panic(fmt.Sprintf("Error: %d, %s", resp.StatusCode, string(body)))
    }

    body, err := ioutil.ReadAll(resp.Body)
    if err != nil {
        panic(err)
    }

    fmt.Println(string(body))
}

func getPromptById(id string) {
    url := fmt.Sprintf("https://swarms.world/get-prompts/%s", id)
    resp, err := http.Get(url)
    if err != nil {
        panic(err)
    }
    defer resp.Body.Close()

    if resp.StatusCode != http.StatusOK {
        body, _ := ioutil.ReadAll(resp.Body)
        panic(fmt.Sprintf("Error: %d, %s", resp.StatusCode, string(body)))
    }

    body, err := ioutil.ReadAll(resp.Body)
    if err != nil {
        panic(err)
    }

    fmt.Println(string(body))
}

func main() {
    filters := map[string]string{
        "name":                 "example",
        "tag":                  "tag1,tag2",
        "use_case":             "example",
        "use_case_description": "description",
    }
    getPrompts(filters)
    getPromptById("123")
}
```

#### Attributes Table

Attribute	Type	Description
id	String	Unique identifier for the prompt
name	String	Name of the prompt
description	String	Description of the prompt
prompt	String	The actual prompt text
use_cases	Array	Use cases for the prompt
tags	String	Tags associated with the prompt

## Additional Information and Tips

- Handle different error statuses appropriately to provide clear feedback to users.
- Consider implementing rate limiting and logging for better security and monitoring.

## References and Resources

- [Next.js API Routes](https://nextjs.org/docs/api-routes/introduction)
- [Supabase Documentation](https://supabase.com/docs)
- [Node Fetch](https://www.npmjs.com/package/node-fetch)
- [Requests Library (Python)](https://docs.python-requests.org/en/latest/)

- [Go net/http Package](https://pkg.go.dev/net/http)

This documentation provides a comprehensive guide to the `getAllPrompts` API endpoint, including usage examples in multiple programming languages and detailed attribute descriptions.