STELLA ONI – SAMPLE WORK

API Overview: Retrieve User Data by Email Address (GET Method)

The “Retrieve User Data by Email Address” API enables you to retrieve user data based on email provided. It returns user id, name, address, phone contact and other relevant user data. The parameter uses email to search for the details of the user and returns a response.

Endpoint: /api/v1/users

The example request is
GET https://api/v1/users?email=janeB@sample.com

The example response is:

{
"id": 2345,
"name": "Jane Brown",
"email": "janeb@sample.com",
“address”: “1223 AnyRoad Lane, Why, UK”,
“phone”: “+44(0000) 000-000”,
"date registered": "2023-06-27",
}

Who would use the API:

- Developers building applications, websites or other systems that require email addresses
- Web developers building a user-friendly interface which requires only email address input from their clients
- Developers creating a personalisation experience
- Business owners using it to retrieve customer data through CRM or something similar.

The API is useful for the following;

- Displaying a user’s information, their profile or for an application
- It is useful for personalising a user’s experience
- It is useful for verifying a user’s identity.
- Customer Support might also use an email address to access data

Conclusion:
The “Retrieve User Data by Email Address” API using the GET method provides a straightforward approach to retrieving user information based on their email addresses. Developers can integrate this API into their applications or systems and, retrieve user data efficiently and securely for various purposes, such as personalization, customer support, or data analysis.

Care should be taken to protect any user data acquired by the application from this endpoint.

API Reference Documentation

Retrieve user data using email addresses

Retrieves the ids, names and other user details using their email addresses

URL

GET https://api/v1/users

Query Parameters

Parameter	Description	Type	Required	Notes
Email	The email address of the user to retrieve the data for	String	Required	Ensure email is in the correct format and matches an existing user

Header Parameter

Header Name	Description	Required	Values
Bearer	The access token for the API authorization	Required	See the authorization section
Accept	The format of the returned data	Optional	application/json or application/xml Default is application/json

Response

Element	Description	Type	Notes
Id	Uniquely identifies the user	integer	
name	The name of the user	string	
email	The email address of the user	string	
address	The address of the user	string	
phone	The phone number of the user	string	
date registered	The date the user registered	string	Format: YYYY-MM-DD

Sample Response:


```
{
 "id": 2345,
  "name": "Jane Brown",
  "email": "janeb@sample.com",
```

“address”: “1223 AnyRoad Lane, Why, UK”,
“phone”: “+44(0000) 000-000”,
"date registered": "2023-06-27",
}

Error codes

Code	Description	Notes
200	Successful	user details was successfully retrieved
400	Bad Request	Returned when the email parameter is missing or was not provided
404	Not found	When the provided email does not exist in the system or has no associated user

Developer Step by Step Guide: Retrieve User Data by Email Address (GET Method)

This guide provides a step-by-step guide on how to use the GET Method to request user information based on their email addresses. It will enable developers to efficiently integrate this to their applications or systems. It is also to ensure that your code is interacting successfully with the API. In addition, that your API keys and authorization work.

Step 1

Signing up for your account

Welcome to the developer documentation for GLFL. It is filled with resources and instructions to assist you in integrating with GLFL APIs and using the platform to its full potential. Contact GLFL's integration team and ask for access to the developer portal to get started. You can set up a team workspace, invite team members, and setup your application to make authenticated API calls once your access has been granted.

You can take the following steps:

1. **Request access to the portal**
   To discuss the needs of your project and to receive access to the Developer Portal, get in touch with our integration team. You will receive the tools and assistance you need from our team to ensure a seamless integration process

2. **Configure your application**
   Set up the OAuth settings in your application to securely access user data and communicate with GLFL APIs. To appropriately configure redirect URLs and other essential OAuth configurations, adhere to our instructions.

3. **Make an authenticated request**
   Set up the OAuth settings in your application to securely access user data and communicate with GLFL APIs. Follow the instructions to correctly configure redirect URLs and other essential OAuth configurations

Getting your API authorization key

Introduction to OAuth
This guide will help you to understand GLFL's OAuth implementation, starting with an introduction to OAuth and the fundamental concepts required to begin the integration process.

What is OAuth?
OAuth (Open Authorization) is a protocol that allows users to share their private information between applications without giving away their passwords. GLFL's APIs use OAuth to ensure a secure experience for merchants wanting to share some portion of their account, like company, store or user details, with another application.

Step 2

Making a request

To make your request with the GET Method based on an email address, follow these steps:

- Input your API endpoint URL and add the email parameter to the URL:

  For example

  GET https://api/v1/users?email={email}

- Replace `{email}` with the user email address.

  For example
  GET /api/v1/users/email?email=JaneB@sample.com

- Add your authorization information in the request header

A response is generated from the API along with error codes and messages.

Step 3

The API Response

From the API response, extract and use the data in your application as required. The response will include information like:

- 'id': The id of the user
- `name`: The name of the user.
- `email`: The email address of the user.
- `address`: The address of the user.
- `phone`: The phone number of the user.
- And more information related to the user.

Response sample code

```
{
        "id": 1,
        "name": "Jane Brown ",
        "email": "JaneB@sample.com",
         "Phone: +44(0000) 000 000
        "created_at": "2023-06-26T12:00:00Z",
        "updated_at": "2023-06-26T12:00:00Z"
    }
```

Error Response

Error response is generated based on particular error codes. A successful response means that the user data has been provided. Errors are generated either through HTTP status

codes which tells you if a request API is successful or has failed. Or in the response body. You can then take applicable action based on this.

Example
200 - successful
400 - bad request
404 - user not found

Sample code
{
“error code”: 404”
“error message: “User not found”
}

Authentication and Security
It is essential that user data is kept secure.
This means using secure protocols like HTTPS and SSL when requesting the user data.
- Ensure you keep your API access credentials private.
- Regularly update your API access information and follow the GLFL guideline.

Rate Limiting
To allow fair usage, you will be allocated a rate limit per month for your usage.

Conclusion
This guide enables you to retrieve user data based on email addresses using the GET method API. Follow it to enable a seamless integration of the functionality into your systems. The aim is to improve user experiences, for personalisation and other functions.

Style Guide Update for GLFL documentation

This style guide offers advice and principles for formatting API documentation.

1. Headings and Sections: Divide the material into sections and give each one distinct, informative heading.

2. Code Samples: Add code snippets to demonstrate how to use API calls and responses. Use a different font for codes to distinguish them from normal text.

3. Consistent style: Provide brief samples of API requests and responses. Also have consistent style for the request bodies, headers, query parameters, and response structures.

4. Tables: Provide columns and rows names for tables that are easy to remember. Clearly label rows and columns for easier understanding and add descriptions.

5. Bold and Italics: Use bold, italics, or other formatting options to draw attention to essential points, significant notes, or unique concerns. This makes important elements more noticeable.

6. Standard naming: Use the same formatting techniques across all of your documents. Maintain standard naming, spacing, and indentation practises. This guarantees a sleek and professional image.

7. Descriptions: Every API endpoint, parameter, and response field should have a clear purpose and function described. Include any constraints, limitations, or restrictions that should be made known to developers.

8. Visual aids: Include visual components to demonstrate the API architecture, processes, or data flows, such as diagrams or flowcharts. Visual representations can improve comprehension and make it easier for developers to use your API.

10. Versioning Information: Clearly state the version number in the documentation if your API supports it.

12. Proofread: To keep your presentation professional, double-check your writing for grammatical, spelling, and formatting mistakes.

You can guarantee consistency, clarity, and usability for developers using GLFL API by adhering to this style guide. To help developers understand and utilise the API successfully, always remember to offer thorough and up-to-date documentation.

By Stella Oni