# Disclaimer: This output contains AI-generated content; user is advised to review it before consumption.  
  
#\*Start of AI Generated Content\*

# **Profile Management API**

## **Introduction**

This document provides a technical overview of the Profile Management API, a Python-based solution for managing user profiles. The API provides two primary functions: updating a user's profile and retrieving a user's profile.

## **Problem Statement**

The problem addressed by this code is the need for a reliable and efficient way to manage user profiles. The API must be able to handle updates to user profiles and retrieve profile information in a secure and scalable manner.

## **Solution Approach**

The solution approach used in this code is to design a RESTful API using Python and the Requests library. The API uses a simple and intuitive endpoint structure, with clear and concise error messaging.

## **Functionality of Code**

The code consists of two primary functions:

### update\_profile(profile\_id, new\_data)

This function updates a user's profile with new data. It takes two arguments:

* profile\_id: The ID of the profile to update
* new\_data: A dictionary containing the new data to update the profile with

The function sends a PUT request to the update profile endpoint with the new data. If the response is successful (200 OK), it returns a success message. Otherwise, it returns an error message.

### get\_profile(profile\_id)

This function retrieves a user's profile. It takes one argument:

* profile\_id: The ID of the profile to retrieve

The function sends a GET request to the profile endpoint. If the response is successful (200 OK), it returns the profile data as a dictionary. Otherwise, it returns None.

## **Input and Output Format**

### Input Format

* profile\_id: An integer representing the ID of the profile to update or retrieve
* new\_data: A dictionary containing the new data to update the profile with (only applicable for update\_profile function)

### Output Format

* update\_profile: A string containing a success message or an error message
* get\_profile: A dictionary containing the profile data or None if the profile is not found

## **Example Usage**

profile\_id = 123  
new\_data = {"name": "John Doe", "email": "johndoe@example.com"}  
  
updated\_profile\_message = update\_profile(profile\_id, new\_data)  
print(updated\_profile\_message)  
  
retrieved\_profile = get\_profile(profile\_id)  
print(retrieved\_profile)

## **Conclusion**

In conclusion, the Profile Management API provides a simple and efficient way to manage user profiles. The API is designed to be scalable and secure, with clear and concise error messaging. The code is well-structured and easy to understand, making it a reliable solution for managing user profiles.

#\*End of AI Generated Content\*