

# Chatgpt Generate Ruby Code to Check User Exists in Github

📅 | 📖 2261 words | 👤 Kracekumar

On saturday night, I want to work on a side project. To find a name for the project and I wanted to create an Github organization in the same name. I started trying out the name one after the other, all the names were taken and thought about writing small script in ruby. Then from nowhere I decided to let chatgpt to write the code for me.

In this blog post, I'll share some of the code generated by chatgpt for the checking whether a profile name exists in the Github with increasing complexity.

You can signup for chatgpt from openai site. (<https://chat.openai.com/chat>)

## Case 1: Simple case



Can you write a ruby program to check a a list of profile names exists in github?



This content may violate our [content policy](#). If you believe this to be in error, please [submit your feedback](#) — your input will aid our research in this area.



Copy code



```
require 'octokit'

# Replace with your access token
ACCESS_TOKEN = "ACCESS_TOKEN"

# Replace with the list of profile names
profiles = ["profile1", "profile2", "profile3"]

client = Octokit::Client.new(access_token:
ACCESS_TOKEN)

profiles.each do |profile|
  user = client.user(profile)
  if user
    puts "#{profile} exists on GitHub"
  else
    puts "#{profile} does not exist on GitHub"
  end
end
end
```

Try again

## Prompt

Can you write a ruby program to **check** a a list of profile names exists in github?

## Generated Code

```

require 'octokit'

# Replace with your access token
ACCESS_TOKEN = "ACCESS_TOKEN"

# Replace with the list of profile names
profiles = ["profile1", "profile2", "profile3"]

client = Octokit::Client.new(access_token: ACCESS_TOKEN)

profiles.each do |profile|
  user = client.user(profile)
  if user
    puts "#{profile} exists on GitHub"
  else
    puts "#{profile} does not exist on GitHub"
  end
end

```

The code looks legit and may run, but there are two issues

- There is no instruction to install `octokit`.
- Not reading the secret from the environment variable is a bad sign.

Let's enhance the instruction by explicitly stating it.

## Prompt

Can you write a ruby program to **check** a a list of **profile names exists in github?**  
**If you use external** gem, specify the installation instruction. Also **read secrets from** environment variable.

## Generated code

```

require 'octokit'

# Replace with the list of profile names
profiles = ["profile1", "profile2", "profile3"]

# Read access token from environment variable
access_token = ENV["ACCESS_TOKEN"]

client = Octokit::Client.new(access_token: access_token)

profiles.each do |profile|
  user = client.user(profile)
  if user
    puts "#{profile} exists on GitHub"
  else
    puts "#{profile} does not exist on GitHub"
  end
end

```

Along with the code, there is an installation line, To install the octokit gem, run `gem install octokit` in your terminal.

## Output

```
$ ruby case_1.rb
To use retry middleware with Faraday v2.0+, install `faraday-retry` gem
profile1 exists on GitHub
profile2 exists on GitHub
profile3 exists on GitHub
```

The code actually works!

Now let's slightly modify the requirements to read list of profile names from a file.

# Case 2: Read list of profile names from a file, check the profiles and write to JSON file

## Prompt

```
Can you write a ruby program to read a list of github profile from a text file,
check each profile exists in github and write the result to a JSON file?
If you use external gem, specify the installation instruction. Also read secrets from environmen
t variable.
```

## Generated code

```
require 'octokit'  
require 'json'  
  
# Read profile names from text file  
profiles = File.readlines("profiles.txt").map(&:chomp)  
  
# Read access token from environment variable  
access_token = ENV["ACCESS_TOKEN"]  
  
client = Octokit::Client.new(access_token: access_token)  
  
# Create a new hash to store the result  
result = {}  
  
profiles.each do |profile|  
  user = client.user(profile)  
  if user  
    result[profile] = true  
  else  
    result[profile] = false  
  end  
end  
  
# Write the result to a JSON file  
File.open("result.json","w") do |f|  
  f.write(result.to_json)  
end
```

## Output

```
$cat profiles.txt
```

```
kracekumar
```

```
zzzzzz
```

```
404
```

```
$ruby case_2.rb
```

```
To use retry middleware with Faraday v2.0+, install `faraday-retry` gem
```

```
Traceback (most recent call last):
```

```
 15: from case_2.rb:15:in `<main>'
 14: from case_2.rb:15:in `each'
 13: from case_2.rb:16:in `block in <main>'
 12: from /Users/krace/.rbenv/versions/2.7.2/lib/ruby/gems/2.7.0/gems/octokit-6.0.1/lib/octokit/client/users.rb:34:in `user'
 11: from /Users/krace/.rbenv/versions/2.7.2/lib/ruby/gems/2.7.0/gems/octokit-6.0.1/lib/octokit/connection.rb:19:in `get'
 10: from /Users/krace/.rbenv/versions/2.7.2/lib/ruby/gems/2.7.0/gems/octokit-6.0.1/lib/octokit/connection.rb:156:in `request'
  9: from /Users/krace/.rbenv/versions/2.7.2/lib/ruby/gems/2.7.0/gems/sawyer-0.9.2/lib/sawyer/agent.rb:99:in `call'
  8: from /Users/krace/.rbenv/versions/2.7.2/lib/ruby/gems/2.7.0/gems/faraday-2.7.1/lib/faraday/connection.rb:200:in `get'
  7: from /Users/krace/.rbenv/versions/2.7.2/lib/ruby/gems/2.7.0/gems/faraday-2.7.1/lib/faraday/connection.rb:445:in `run_request'
  6: from /Users/krace/.rbenv/versions/2.7.2/lib/ruby/gems/2.7.0/gems/faraday-2.7.1/lib/faraday/rack_builder.rb:153:in `build_response'
  5: from /Users/krace/.rbenv/versions/2.7.2/lib/ruby/gems/2.7.0/gems/octokit-6.0.1/lib/octokit/middleware/follow_redirects.rb:61:in `call'
  4: from /Users/krace/.rbenv/versions/2.7.2/lib/ruby/gems/2.7.0/gems/octokit-6.0.1/lib/octokit/middleware/follow_redirects.rb:73:in `perform_with_redirection'
  3: from /Users/krace/.rbenv/versions/2.7.2/lib/ruby/gems/2.7.0/gems/faraday-2.7.1/lib/faraday/middleware.rb:17:in `call'
  2: from /Users/krace/.rbenv/versions/2.7.2/lib/ruby/gems/2.7.0/gems/faraday-2.7.1/lib/faraday/response.rb:42:in `on_complete'
  1: from /Users/krace/.rbenv/versions/2.7.2/lib/ruby/gems/2.7.0/gems/faraday-2.7.1/lib/faraday/middleware.rb:18:in `block in call'
/Users/krace/.rbenv/versions/2.7.2/lib/ruby/gems/2.7.0/gems/octokit-6.0.1/lib/octokit/response/raise_error.rb:14:in `on_complete': GET https://api.github.com/users/404: 404 - Not Found // See: https://docs.github.com/rest/reference/users#get-a-user (Octokit::NotFound)
```

Ah ha. the profile 404 returns 404 and the program fails, now enhance the prompt to modify it.

## Prompt

Can you write a ruby program to read a list of github profile from a text file, **check each profile exists in github and write** the result to a JSON file? The code should handle exceptions.

**If you use external** gem, specify the installation instruction. Also **read secrets from** environment variable.

## Generated code

```

require 'octokit'
require 'json'

# Read profile names from text file
profiles = File.readlines("profiles.txt").map(&:chomp)

# Read access token from environment variable
access_token = ENV["ACCESS_TOKEN"]

client = Octokit::Client.new(access_token: access_token)

# Create a new hash to store the result
result = {}

profiles.each do |profile|
  begin
    user = client.user(profile)
    if user
      result[profile] = true
    else
      result[profile] = false
    end
  rescue Octokit::NotFound
    result[profile] = false
  end
end

# Write the result to a JSON file
File.open("result.json", "w") do |f|
  f.write(result.to_json)
end

```

- Now the model generates the code with exception handling but only handles 404 without explicitly mentioning the specific exception in the prompt. Neat!

## Output

```

$ruby case_2.rb
To use retry middleware with Faraday v2.0+, install `faraday-retry` gem
$cat result.json
{"kracekumar":true,"zzzzz":true,"404":false}

```

# Case 3: Read from dictionary, check the profile in github, handle rate-limit and write the results to file.

## Prompt

Can you write a ruby program to read a list of github profile from a text file in the location /usr/share/dict/words,

**check each profile exists in github, check for rate limit and write the result to a JSON file?** Print result of each user check in the terminal. The code should handle exceptions.

**If you use external gem, specify the installation instruction. Also read secrets from environment variable.**

## Generated Code

```
require 'octokit'
require 'json'

# Read profile names from text file
profiles = File.readlines("/usr/share/dict/words").map(&:chomp)

# Read access token from environment variable
access_token = ENV["ACCESS_TOKEN"]

client = Octokit::Client.new(access_token: access_token)

# Create a new hash to store the result
result = {}

profiles.each do |profile|
  # Check for rate limit
  rate_limit = client.rate_limit
  if rate_limit.remaining == 0
    puts "Rate limit exceeded. Waiting for reset at #{rate_limit.reset_at}..."
    sleep rate_limit.resets_in
  end

  begin
    user = client.user(profile)
    if user
      result[profile] = true
      puts "#{profile} exists on GitHub"
    else
      result[profile] = false
      puts "#{profile} does not exist on GitHub"
    end
  rescue Octokit::NotFound
    result[profile] = false
    puts "#{profile} does not exist on GitHub"
  end
end

# Write the result to a JSON file
File.open("result.json", "w") do |f|
  f.write(result.to_json)
end
```

## Output



```
$ruby case_3.rb
To use retry middleware with Faraday v2.0+, install `faraday-retry` gem
A exists on GitHub
a exists on GitHub
aa exists on GitHub
aal exists on GitHub
...
abarthrosis does not exist on GitHub
Traceback (most recent call last):
  2: from case_3.rb:15:in `'
  1: from case_3.rb:15:in `each'
case_3.rb:19:in `block in <main>': undefined method `reset_at' for #<Octokit::RateLimit:0x00000000107ba8760> (NoMethodError)
Did you mean?  resets_at
               resets_at=
```

Oops, the code breaks at line number 19 - `rate_limit.reset_at`.

Let's give feedback to the chatgpt, saying the code fails at line 19.

## Prompt

```
the above code failed with the following error, undefined method `reset_at' for #<Octokit::RateLimit:0x0000000010498f210> (NoMethodError)
Did you mean?  resets_at
               resets_at=

Can you fix it?
```

## Generated code

```

require 'octokit'
require 'json'

# Read profile names from text file
profiles = File.readlines("/usr/share/dict/words").map(&:chomp)

# Read access token from environment variable
access_token = ENV["ACCESS_TOKEN"]

client = Octokit::Client.new(access_token: access_token)

# Create a new hash to store the result
result = {}

profiles.each do |profile|
  # Check for rate limit
  rate_limit = client.rate_limit
  if rate_limit.remaining == 0
    puts "Rate limit exceeded. Waiting for reset at #{rate_limit.resets_at}..."
    sleep rate_limit.resets_in
  end

  begin
    user = client.user(profile)
    if user
      result[profile] = true
      puts "#{profile} exists on GitHub"
    else
      result[profile] = false
      puts "#{profile} does not exist on GitHub"
    end
  rescue Octokit::NotFound
    result[profile] = false
    puts "#{profile} does not exist on GitHub"
  end
end

# Write the result to a JSON file
File.open("result.json","w") do |f|
  f.write(result.to_json)
end

```

The model generated above code with a message, The error occurred because the Octokit::RateLimit object does not have a method named reset\_at. The correct method is resets\_at. Great!

## Output

```

$ruby case_3.rb
To use retry middleware with Faraday v2.0+, install `faraday-retry` gem
Rate limit exceeded. Waiting for reset at 2022-12-04 19:46:29 +0000...

```

Now the bug is fixed and worked as expected! I don't say issues. OF course! there can be all sort of exceptions that need to be handled - socket timeout, slow response time, 500s etc... Saving the results to a file after checking all profiles is problematic.

## Case 4: Let's tell the model to refactor the code into functions.

### Prompt

```
Thanks, that fixed exception. Can you refactor the code into functions?
```

### Generated Code

```

require 'octokit'
require 'json'

# Read profile names from text file
def read_profiles
  File.readlines("/usr/share/dict/words").map(&:chomp)
end

# Check if user exists on GitHub
def check_user(user)
  begin
    client.user(user)
    true
  rescue Octokit::NotFound
    false
  end
end

# Check for rate limit and wait if necessary
def check_rate_limit
  rate_limit = client.rate_limit
  if rate_limit.remaining == 0
    puts "Rate limit exceeded. Waiting for reset at #{rate_limit.resets_at}..."
    sleep rate_limit.resets_in
  end
end

# Read access token from environment variable
access_token = ENV["ACCESS_TOKEN"]

client = Octokit::Client.new(access_token: access_token)

# Create a new hash to store the result
result = {}

profiles = read_profiles

profiles.each do |profile|
  check_rate_limit

  result[profile] = check_user(profile)
  puts "#{profile} exists on GitHub" if result[profile]
  puts "#{profile} does not exist on GitHub" unless result[profile]
end

# Write the result to a JSON file
File.open("result.json", "w") do |f|
  f.write(result.to_json)
end

```

The generated code is syntactically invalid, since `client` variable is undefined in 21

**Prompt**

I get the error when I run the above code, Traceback (most recent **call last**):

```
3: from case_3.rb:38:in `'
```

```
2: from case_3.rb:38:in `each'
```

```
1: from case_3.rb:39:in `block in <main>'
```

```
case_3.rb:21:in `check_rate_limit': undefined local variable or method `client' for main:Object  
(NameError)
```

## Generated Code

```

require 'octokit'
require 'json'

# Read profile names from text file
def read_profiles
  File.readlines("/usr/share/dict/words").map(&:chomp)
end

# Check if user exists on GitHub
def check_user(client, user)
  begin
    client.user(user)
    true
  rescue Octokit::NotFound
    false
  end
end

# Check for rate limit and wait if necessary
def check_rate_limit(client)
  rate_limit = client.rate_limit
  if rate_limit.remaining == 0
    puts "Rate limit exceeded. Waiting for reset at #{rate_limit.resets_at}..."
    sleep rate_limit.resets_in
  end
end

# Read access token from environment variable
access_token = ENV["ACCESS_TOKEN"]

client = Octokit::Client.new(access_token: access_token)

# Create a new hash to store the result
result = {}

profiles = read_profiles

profiles.each do |profile|
  check_rate_limit(client)

  result[profile] = check_user(client, profile)
  puts "#{profile} exists on GitHub" if result[profile]
  puts "#{profile} does not exist on GitHub" unless result[profile]
end

# Write the result to a JSON file
File.open("result.json", "w") do |f|
  f.write(result.to_json)
end

```

Fix message is better: The error occurred because the `check_rate_limit` and `check_user` functions did not have access to the client object. To fix the error, the client object must be passed as an argument to these functions.

## Output

```
$ruby case_3.rb
```

```
To use retry middleware with Faraday v2.0+, install `faraday-retry` gem
```

```
Rate limit exceeded. Waiting for reset at 2022-12-04 19:46:29 +0000...
```

# Case 5: Write the test cases

## Prompt

```
Can you write the test cases for the above code?
```

## Generated Code

```

require 'minitest/autorun'
require 'octokit'
require 'json'

class TestGitHubProfileChecker < Minitest::Test
  def setup
    @access_token = ENV["ACCESS_TOKEN"]
    @client = Octokit::Client.new(access_token: @access_token)
  end

  def test_read_profiles
    expected_profiles = ["test1", "test2"]
    File.stub :readlines, expected_profiles do
      assert_equal expected_profiles, read_profiles
    end
  end

  def test_check_user_exists
    user = "octocat"
    assert check_user(@client, user)
  end

  def test_check_user_does_not_exist
    user = "not_a_real_user"
    refute check_user(@client, user)
  end

  def test_check_rate_limit
    rate_limit = @client.rate_limit
    rate_limit.stub :remaining, 0 do
      rate_limit.stub :resets_at, Time.now do
        rate_limit.stub :resets_in, 0 do
          assert_output "Rate limit exceeded. Waiting for reset at #{rate_limit.resets_at}...\n" do
            check_rate_limit(@client)
          end
        end
      end
    end
  end
end
end
end
end

```

After adding the previously generated code, a test case failed.



```
ruby test_case.rb
To use retry middleware with Faraday v2.0+, install `faraday-retry` gem
Run options: --seed 34383
```

```
# Running:
```

```
..F.
```

```
Finished in 0.782971s, 5.1087 runs/s, 5.1087 assertions/s.
```

```
1) Failure:
```

```
TestGitHubProfileChecker#test_check_rate_limit [test_case.rb:83]:
```

```
In stdout.
```

```
--- expected
```

```
+++ actual
```

```
@@ -1,2 +1 @@
```

```
-"Rate limit exceeded. Waiting for reset at 2022-12-04 20:24:05 +0000...
```

```
-"
```

```
+""
```

```
4 runs, 4 assertions, 1 failures, 0 errors, 0 skips
```

Looking at the code, rate limit code was stubbed but doesn't behave as expected. Again updating the prompt, a couple of times, the model produces same test code and it is unable to fix it. After multiple prompts, the model stop generating complete code.

## Final generated code

```

require 'minitest/autorun'
require 'octokit'
require 'json'

class TestGitHubProfileChecker < Minitest::Test
  def setup
    @access_token = ENV["ACCESS_TOKEN"]
    @client = Octokit::Client.new(access_token: @access_token)
  end

  def test_read_profiles
    expected_profiles = ["test1", "test2"]
    File.stub :readlines, expected_profiles do
      assert_equal expected_profiles, read_profiles
    end
  end

  def test_check_user_exists
    user = "octocat"
    assert check_user(@client, user)
  end

  def test_check_user_does_not_exist
    user = "not_a_real_user"
    refute check_user(@client, user)
  end

  def test_check_rate_limit
    now = Time.now
    rate_limit = Octokit::RateLimit.new(remaining: 0, resets_at: now.to_i, resets_in: 0)
    @client.stub :rate_limit, rate_limit do
      assert_output "Rate limit exceeded. Waiting for reset at #{now}...\n" do
        check_rate_limit(@client)
      end
    end
  end

```

## Final conversation

Me: How many lines **of** code are you allowed to generate **in** a given time?

Model: There **is** no specific limit on the number **of** lines **of** code that can be generated **in** a given time.

The quality **and** accuracy **of** the code **is** more important than the quantity **of** code generated.

Me: But your previous code generation was incomplete. can you tell me what happened?

Model: I apologize **if** my previous code generation was incomplete. There can be a number **of** reasons why **this** may have happened.

For example, the prompt may **not** have provided enough information to accurately generate the code, **or** there may have been a technical issue with the code generation system. In any case, I will **do** my best to provide complete **and** accurate code **in** response to future prompts.

## Takeway

- Given enough context and iteration the model is able to correct the simple errors.
- It takes many attempts to get something working, but it's far away from production quality code (could be pure model safe-guard).

- It's able to remember the previous context and improvise based on feedback.
- Once things get slightly complicated the model fails to generate syntactically correct code.

It was definitely fun to play around with the model but making the model to produce fool proof code seems very far away.

Tags: #chatgpt (<https://kracekumar.com//tags/chatgpt/>) #AI (<https://kracekumar.com//tags/ai/>) #language\_model ([https://kracekumar.com//tags/language\\_model/](https://kracekumar.com//tags/language_model/))



(<http://creativecommons.org/licenses/by-sa/4.0/>)

This work is licensed under a Creative Commons Attribution-ShareAlike 4.0 International License (<http://creativecommons.org/licenses/by-sa/4.0/>).

← **PREVIOUS POST ([HTTPS://KRACEKUMAR.COM/POST/MICRO-BENCHMARK-PYTHON-311/](https://kracekumar.com/post/micro-benchmark-python-311/))**

Enter your email

Subscribe

Powered by Buttondown. (<https://buttondown.email>)



(<mailto:kracethekingmaker@gmail.com>)



(<https://github.com/kracekumar>)



(<https://gitlab.com/kracekumar>)



(<https://twitter.com/kracetheking>)



()

Kracekumar • © 2022 • Technical Ramblings (<https://kracekumar.com/>)

Hugo v0.107.0 (<https://gohugo.io>) powered • Theme Beautiful Hugo (<https://github.com/halogenica/beautifulhugo>)  
adapted from Beautiful Jekyll (<https://deanattali.com/beautiful-jekyll/>)