

Premium

Test Report

Test Session Summary

|  |  |
| --- | --- |
| Start: Mon Apr 13 07:45:18 2020 End: Mon Apr 13 07:45:32 2020 Duration: 0:00:14.728000  broken: 1 (14.29%) failed: 0 (0.00%) skipped: 0 (0.00%) passed: 6 (85.71%) |  |

Test Results

# change to premium account - broken

change to premium account

## Details

SystemExit: Testing failed to make claim premium

|  |
| --- |
| self = <Web\_Testing.Pages.premium.PremiumPage object at 0x106612070>   def check\_claim\_premium(self):  try:  self.click\_get\_premium\_button\_1() > alert = self.driver.switch\_to.alert  Web\_Testing/Pages/premium.py:164:  \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_   self = <selenium.webdriver.remote.switch\_to.SwitchTo object at 0x1066125b0>   @property  def alert(self):  """  Switches focus to an alert on the page.    :Usage:  alert = driver.switch\_to.alert  """  alert = Alert(self.\_driver) > alert.text  venv/lib/python3.8/site-packages/selenium/webdriver/remote/switch\_to.py:55:  \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_   self = <selenium.webdriver.common.alert.Alert object at 0x1079ac1c0>   @property  def text(self):  """  Gets the text of the Alert.  """  if self.driver.w3c: > return self.driver.execute(Command.W3C\_GET\_ALERT\_TEXT)["value"]  venv/lib/python3.8/site-packages/selenium/webdriver/common/alert.py:67:  \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_   self = <selenium.webdriver.chrome.webdriver.WebDriver (session="4e6ead03b8937527e4f2d10241e39a89")> driver\_command = 'w3cGetAlertText', params = {}   def execute(self, driver\_command, params=None):  """  Sends a command to be executed by a command.CommandExecutor.    :Args:  - driver\_command: The name of the command to execute as a string.  - params: A dictionary of named parameters to send with the command.    :Returns:  The command's JSON response loaded into a dictionary object.  """  if self.session\_id is not None:  if not params:  params = {'sessionId': self.session\_id}  elif 'sessionId' not in params:  params['sessionId'] = self.session\_id    params = self.\_wrap\_value(params)  response = self.command\_executor.execute(driver\_command, params)  if response: > self.error\_handler.check\_response(response)  venv/lib/python3.8/site-packages/selenium/webdriver/remote/webdriver.py:321:  \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_   self = <selenium.webdriver.remote.errorhandler.ErrorHandler object at 0x10792a1f0> response = {'status': 404, 'value': '{"value":{"error":"no such alert","message":"no such alert\\n (Session info: chrome=80.0.39...fff73025e65 \_pthread\_start + 148\\n19 libsystem\_pthread.dylib 0x00007fff7302183b thread\_start + 15\\n"}}'}   def check\_response(self, response):  """  Checks that a JSON response from the WebDriver does not have an error.    :Args:  - response - The JSON response from the WebDriver server as a dictionary  object.    :Raises: If the response contains an error message.  """  status = response.get('status', None)  if status is None or status == ErrorCode.SUCCESS:  return  value = None  message = response.get("message", "")  screen = response.get("screen", "")  stacktrace = None  if isinstance(status, int):  value\_json = response.get('value', None)  if value\_json and isinstance(value\_json, basestring):  import json  try:  value = json.loads(value\_json)  if len(value.keys()) == 1:  value = value['value']  status = value.get('error', None)  if status is None:  status = value["status"]  message = value["value"]  if not isinstance(message, basestring):  value = message  message = message.get('message')  else:  message = value.get('message', None)  except ValueError:  pass    exception\_class = ErrorInResponseException  if status in ErrorCode.NO\_SUCH\_ELEMENT:  exception\_class = NoSuchElementException  elif status in ErrorCode.NO\_SUCH\_FRAME:  exception\_class = NoSuchFrameException  elif status in ErrorCode.NO\_SUCH\_WINDOW:  exception\_class = NoSuchWindowException  elif status in ErrorCode.STALE\_ELEMENT\_REFERENCE:  exception\_class = StaleElementReferenceException  elif status in ErrorCode.ELEMENT\_NOT\_VISIBLE:  exception\_class = ElementNotVisibleException  elif status in ErrorCode.INVALID\_ELEMENT\_STATE:  exception\_class = InvalidElementStateException  elif status in ErrorCode.INVALID\_SELECTOR \  or status in ErrorCode.INVALID\_XPATH\_SELECTOR \  or status in ErrorCode.INVALID\_XPATH\_SELECTOR\_RETURN\_TYPER:  exception\_class = InvalidSelectorException  elif status in ErrorCode.ELEMENT\_IS\_NOT\_SELECTABLE:  exception\_class = ElementNotSelectableException  elif status in ErrorCode.ELEMENT\_NOT\_INTERACTABLE:  exception\_class = ElementNotInteractableException  elif status in ErrorCode.INVALID\_COOKIE\_DOMAIN:  exception\_class = InvalidCookieDomainException  elif status in ErrorCode.UNABLE\_TO\_SET\_COOKIE:  exception\_class = UnableToSetCookieException  elif status in ErrorCode.TIMEOUT:  exception\_class = TimeoutException  elif status in ErrorCode.SCRIPT\_TIMEOUT:  exception\_class = TimeoutException  elif status in ErrorCode.UNKNOWN\_ERROR:  exception\_class = WebDriverException  elif status in ErrorCode.UNEXPECTED\_ALERT\_OPEN:  exception\_class = UnexpectedAlertPresentException  elif status in ErrorCode.NO\_ALERT\_OPEN:  exception\_class = NoAlertPresentException  elif status in ErrorCode.IME\_NOT\_AVAILABLE:  exception\_class = ImeNotAvailableException  elif status in ErrorCode.IME\_ENGINE\_ACTIVATION\_FAILED:  exception\_class = ImeActivationFailedException  elif status in ErrorCode.MOVE\_TARGET\_OUT\_OF\_BOUNDS:  exception\_class = MoveTargetOutOfBoundsException  elif status in ErrorCode.JAVASCRIPT\_ERROR:  exception\_class = JavascriptException  elif status in ErrorCode.SESSION\_NOT\_CREATED:  exception\_class = SessionNotCreatedException  elif status in ErrorCode.INVALID\_ARGUMENT:  exception\_class = InvalidArgumentException  elif status in ErrorCode.NO\_SUCH\_COOKIE:  exception\_class = NoSuchCookieException  elif status in ErrorCode.UNABLE\_TO\_CAPTURE\_SCREEN:  exception\_class = ScreenshotException  elif status in ErrorCode.ELEMENT\_CLICK\_INTERCEPTED:  exception\_class = ElementClickInterceptedException  elif status in ErrorCode.INSECURE\_CERTIFICATE:  exception\_class = InsecureCertificateException  elif status in ErrorCode.INVALID\_COORDINATES:  exception\_class = InvalidCoordinatesException  elif status in ErrorCode.INVALID\_SESSION\_ID:  exception\_class = InvalidSessionIdException  elif status in ErrorCode.UNKNOWN\_METHOD:  exception\_class = UnknownMethodException  else:  exception\_class = WebDriverException  if value == '' or value is None:  value = response['value']  if isinstance(value, basestring):  if exception\_class == ErrorInResponseException:  raise exception\_class(response, value)  raise exception\_class(value)  if message == "" and 'message' in value:  message = value['message']    screen = None  if 'screen' in value:  screen = value['screen']    stacktrace = None  if 'stackTrace' in value and value['stackTrace']:  stacktrace = []  try:  for frame in value['stackTrace']:  line = self.\_value\_or\_default(frame, 'lineNumber', '')  file = self.\_value\_or\_default(frame, 'fileName', '<anonymous>')  if line:  file = "%s:%s" % (file, line)  meth = self.\_value\_or\_default(frame, 'methodName', '<anonymous>')  if 'className' in frame:  meth = "%s.%s" % (frame['className'], meth)  msg = " at %s (%s)"  msg = msg % (meth, file)  stacktrace.append(msg)  except TypeError:  pass  if exception\_class == ErrorInResponseException:  raise exception\_class(response, message)  elif exception\_class == UnexpectedAlertPresentException:  alert\_text = None  if 'data' in value:  alert\_text = value['data'].get('text')  elif 'alert' in value:  alert\_text = value['alert'].get('text')  raise exception\_class(message, screen, stacktrace, alert\_text) > raise exception\_class(message, screen, stacktrace) E selenium.common.exceptions.NoAlertPresentException: Message: no such alert E (Session info: chrome=80.0.3987.163)  venv/lib/python3.8/site-packages/selenium/webdriver/remote/errorhandler.py:242: NoAlertPresentException  During handling of the above exception, another exception occurred:  self = <Web\_Testing.Tests.test\_premium.TestPremium object at 0x1079acbe0> setup\_begin = None   @allure.severity(allure.severity\_level.CRITICAL)  @allure.story("Premium tests")  @allure.sub\_suite("change to premium account")  @allure.title("change to premium account")  @allure.description("change to premium account")  @pytest.mark.Do  @pytest.mark.Premium  def test\_case\_1(self, setup\_begin): > if self.pp.check\_claim\_premium():  Web\_Testing/Tests/test\_premium.py:78:  \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_  Web\_Testing/Pages/premium.py:174: in check\_claim\_premium  exit('Testing failed to make claim premium') \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_   self = Use exit() or Ctrl-D (i.e. EOF) to exit code = 'Testing failed to make claim premium'   def \_\_call\_\_(self, code=None):  # Shells like IDLE catch the SystemExit, but listen when their  # stdin wrapper is closed.  try:  sys.stdin.close()  except:  pass > raise SystemExit(code) E SystemExit: Testing failed to make claim premium  /Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/\_sitebuiltins.py:26: SystemExit |

## Test Setup

[Fixture] setup\_begin

## Test Body

No test body information available.

## Test Teardown

[Fixture] setup\_begin::0

# Clicking Download link - passed

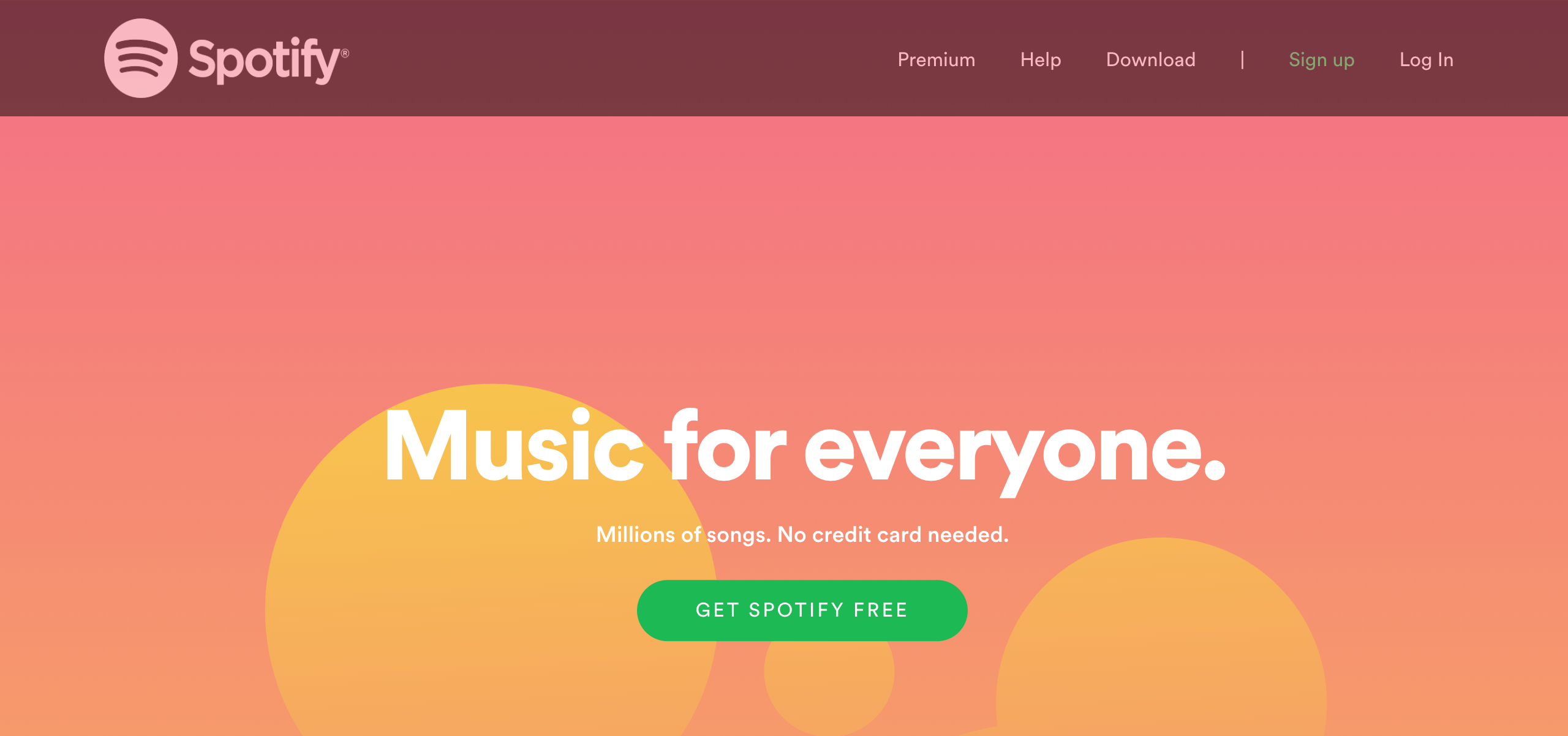
Clicking Download link

## Test Setup

[Fixture] setup

## Test Body

[Attachment] Download link functional



## Test Teardown

[Fixture] setup::0

# Clicking account link - passed

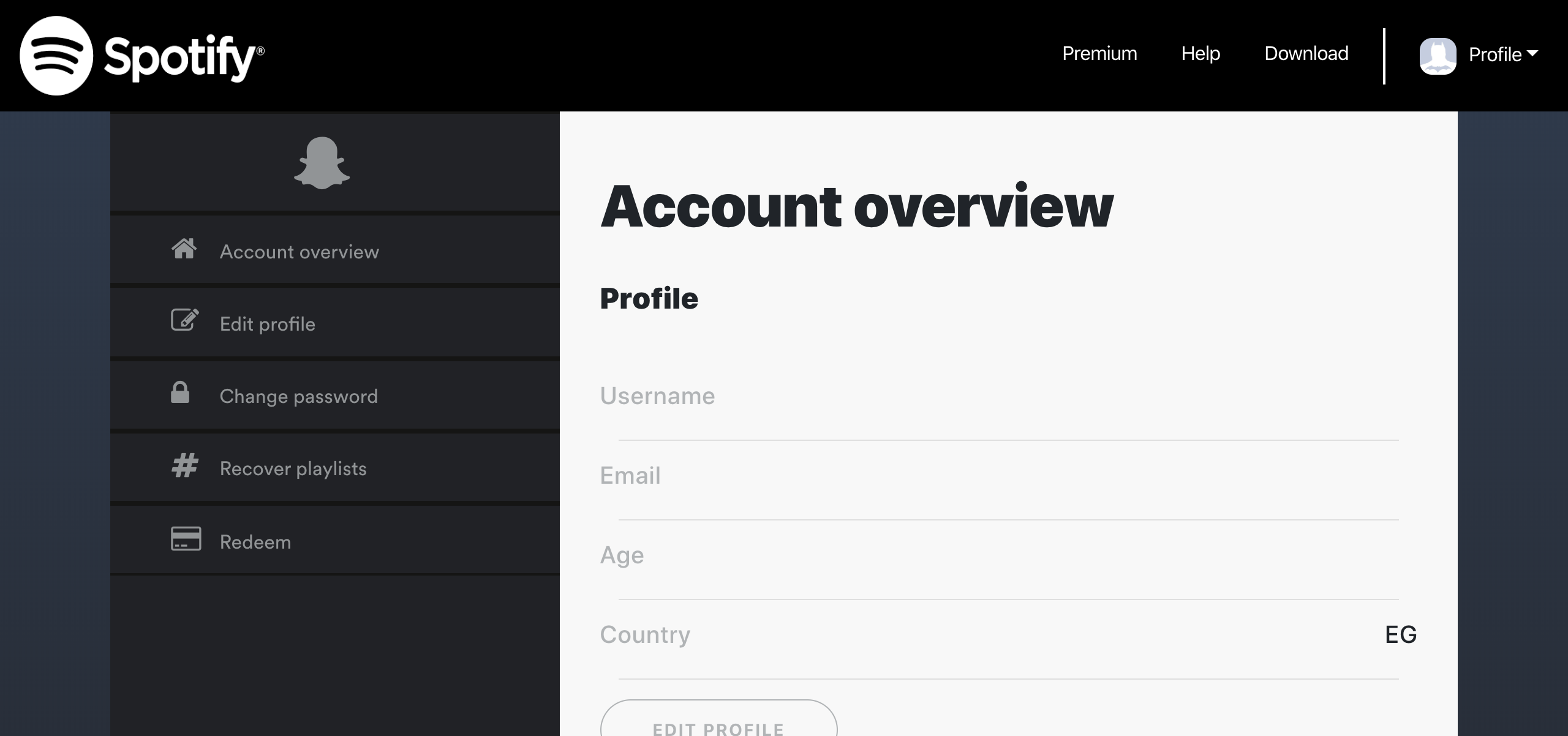
Clicking account link

## Test Setup

[Fixture] setup

## Test Body

[Attachment] Account link functional



## Test Teardown

[Fixture] setup::0

# Clicking help link - passed

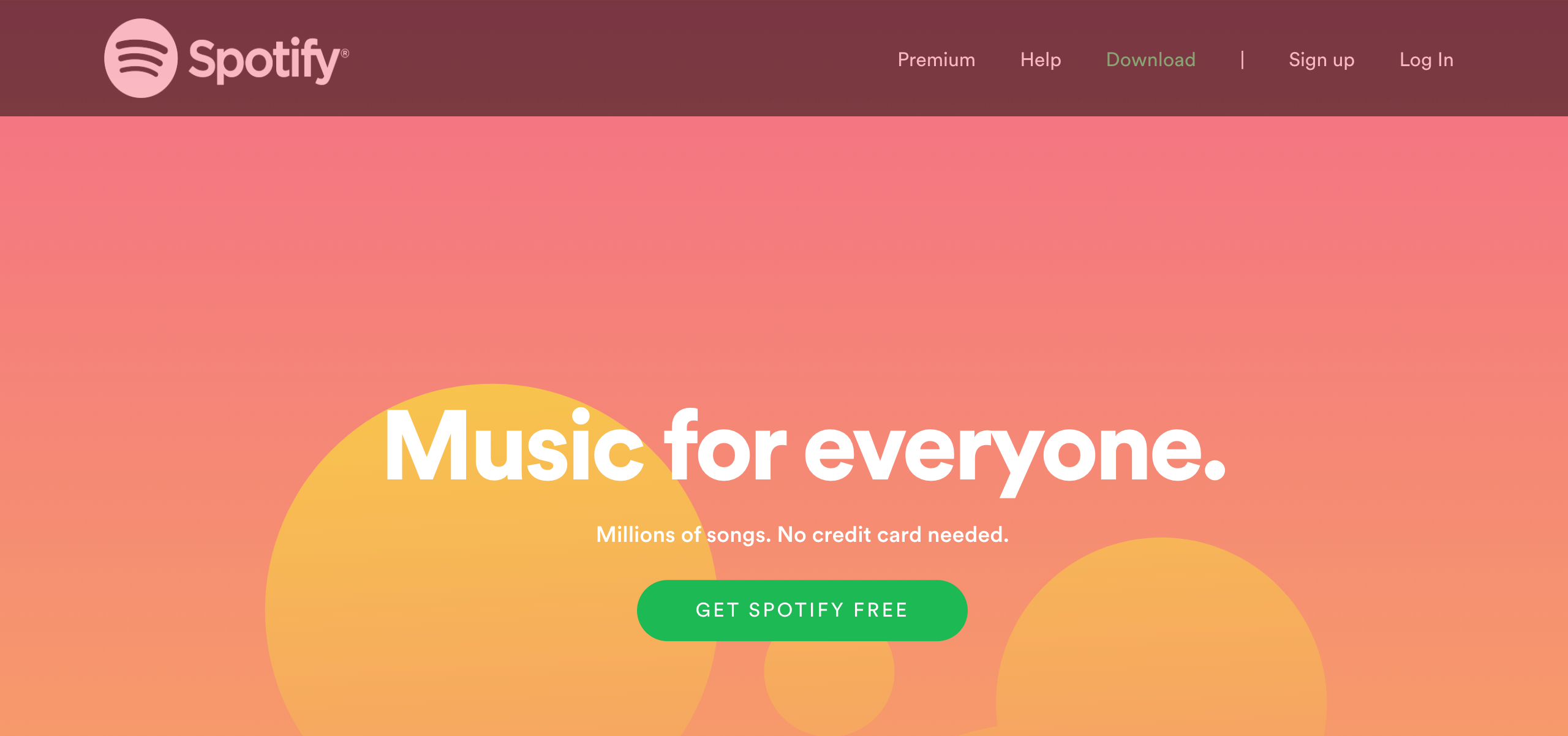
Clicking help link

## Test Setup

[Fixture] setup

## Test Body

[Attachment] Help link functional



## Test Teardown

[Fixture] setup::0

# Clicking logout link - passed

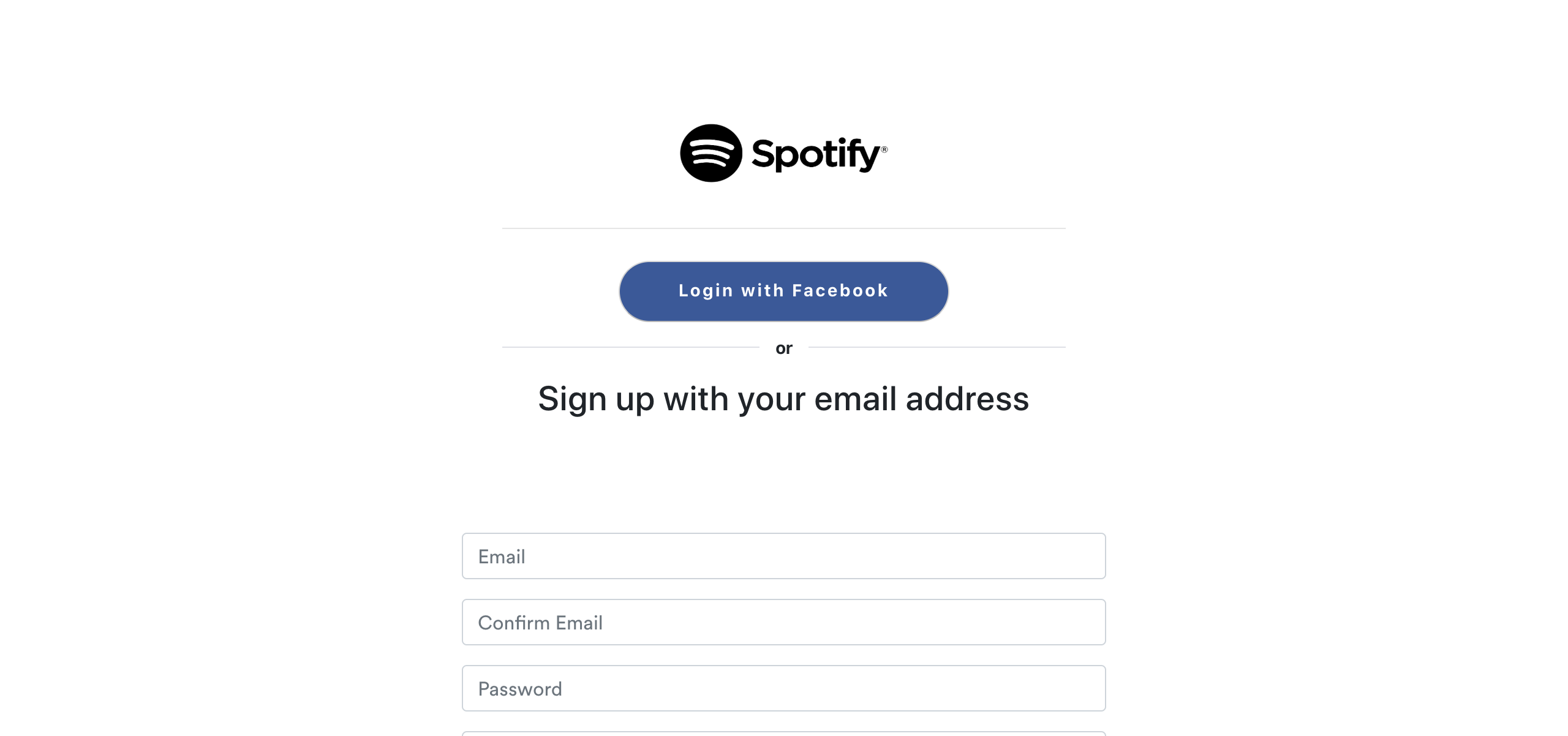
Clicking logout link

## Test Setup

[Fixture] setup\_final

## Test Body

[Attachment] Logout button functional



## Test Teardown

[Fixture] setup\_final::0

# Clicking premium link - passed

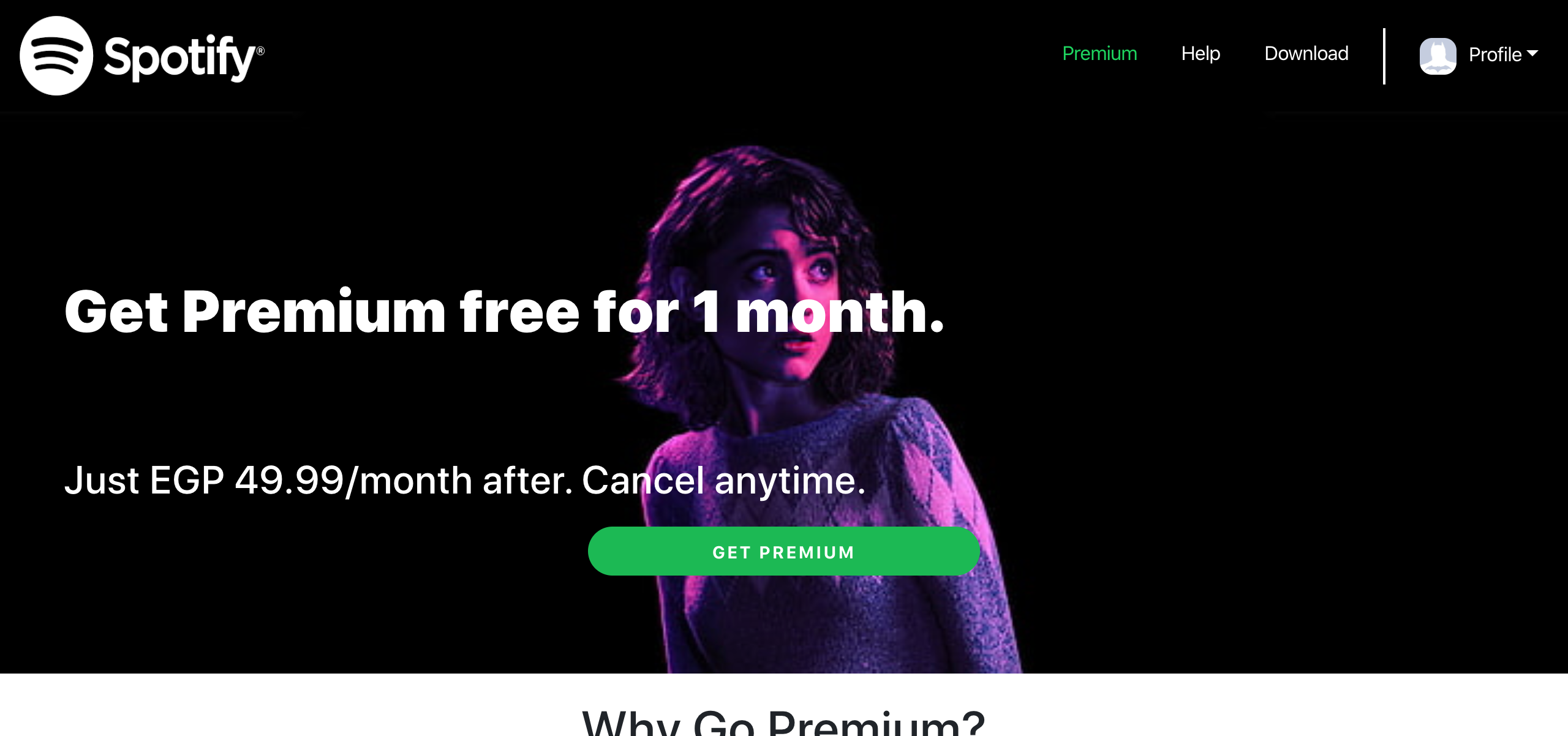
Clicking premium link

## Test Setup

[Fixture] setup

## Test Body

[Attachment] Premium link functional



## Test Teardown

[Fixture] setup::0

# Spotify logo - passed

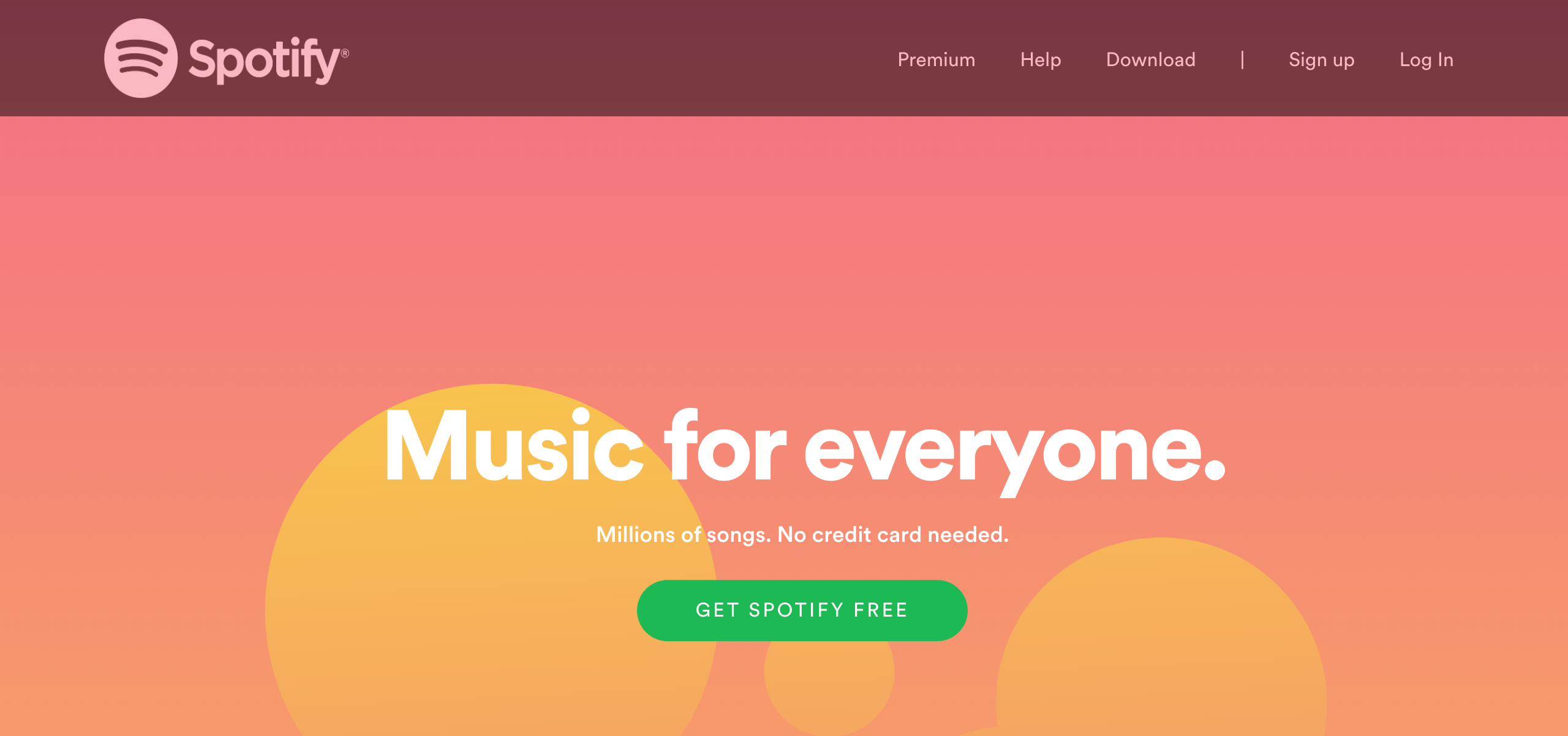
Spotify logo

## Test Setup

[Fixture] setup

## Test Body

[Attachment] Spotify logo functional



## Test Teardown

[Fixture] setup::0