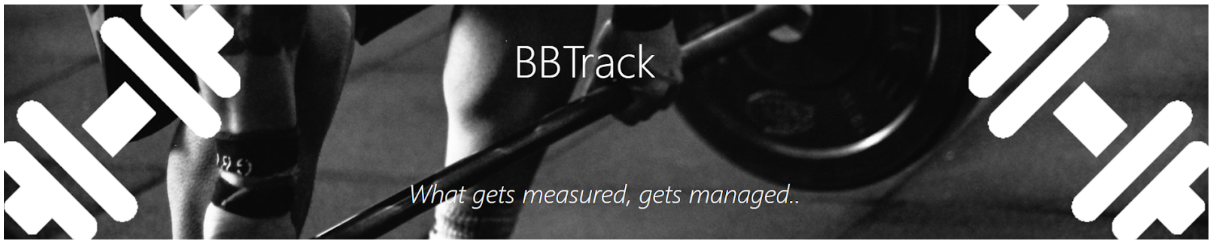
Testing Manual for Barbell Track



Date	06/05/2021
Jordan Voss	17327513
Nathan Ndombasi	13517227

Table of Contents

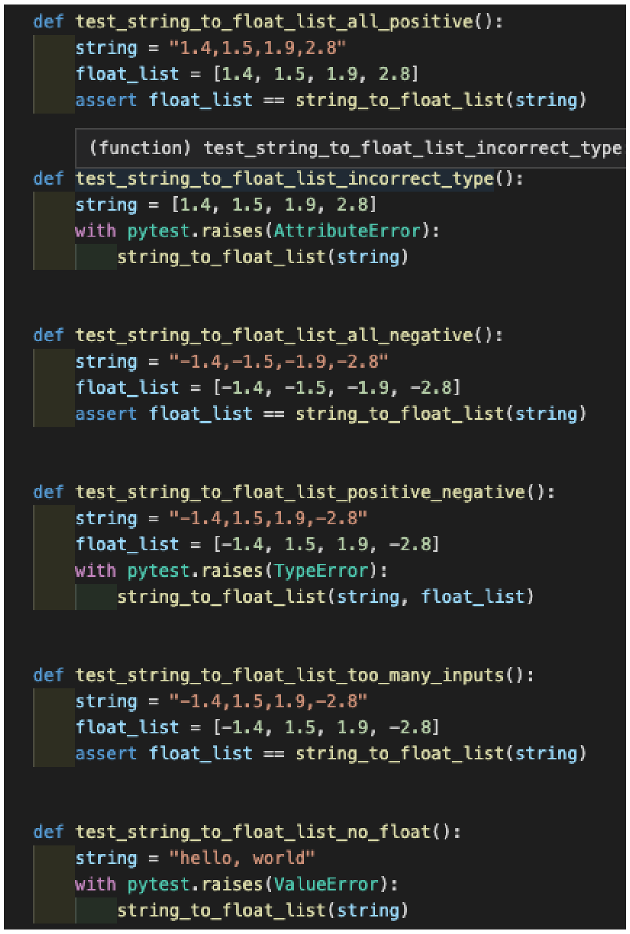
1. Unit Tests

1.1. Function Tests

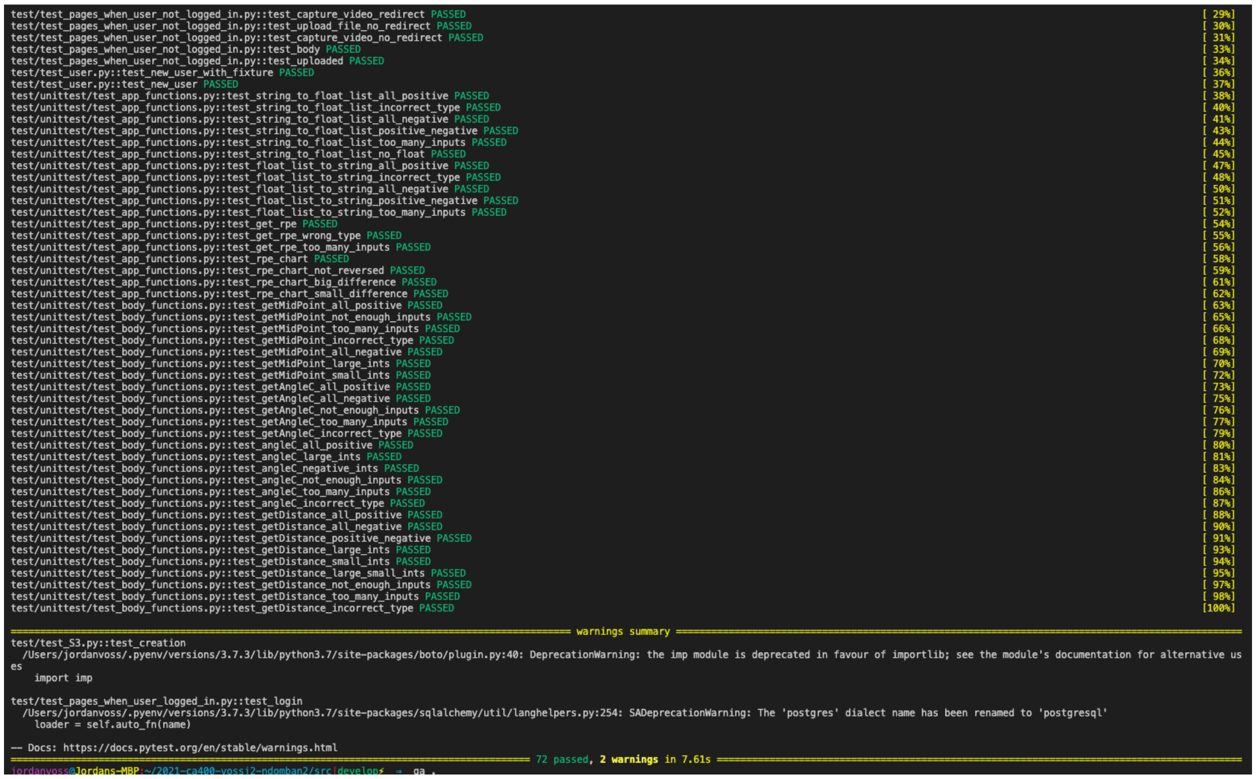
We decided to use pytest to incorporate our unit testing into the app. Our backend was written in python so pytest was perfect for testing the flask pages and functions used when analysing the body parts.

Our strategy was to test that the functions output the correct results when given different inputs, from smaller numbers to larger ones, and various types, that they would accept the right amount of inputs, and both positive and negative numbers. As we were using these functions to analyse body part movement and positioning it was important to validate that these functions would work as we expected them to.

We had in total 72 unit tests to test the core functionality of our backend. An example of some unit tests we wrote are shown below, and the results of them are shown on the following pages.



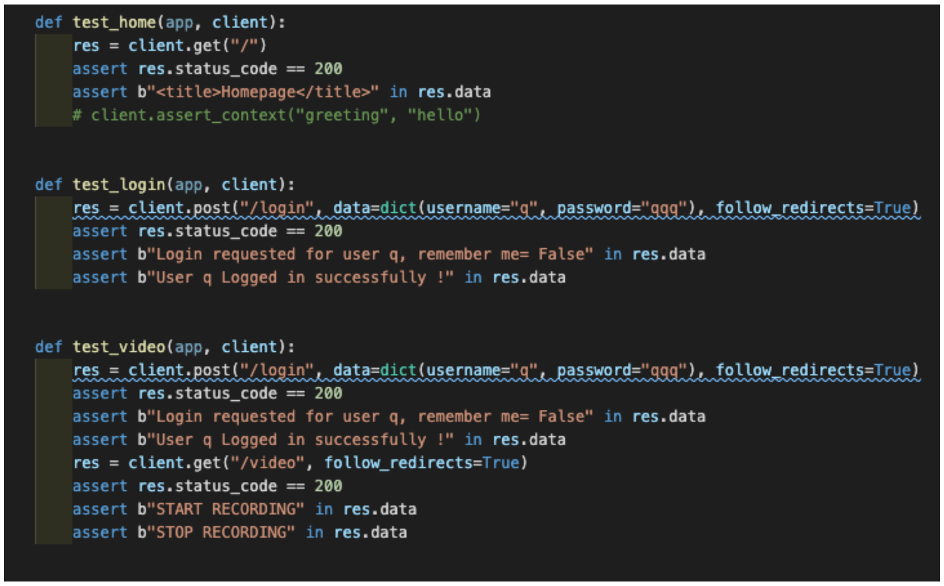
```python
def test_string_to_float_list_all_positive():
    string = "1.4,1.5,1.9,2.8"
    float_list = [1.4, 1.5, 1.9, 2.8]
    assert float_list == string_to_float_list(string)

    (function) test_string_to_float_list_incorrect_type
def test_string_to_float_list_incorrect_type():
    string = [1.4, 1.5, 1.9, 2.8]
    with pytest.raises(AttributeError):
        string_to_float_list(string)


def test_string_to_float_list_all_negative():
    string = "-1.4,-1.5,-1.9,-2.8"
    float_list = [-1.4, -1.5, -1.9, -2.8]
    assert float_list == string_to_float_list(string)


def test_string_to_float_list_positive_negative():
    string = "-1.4,1.5,1.9,-2.8"
    float_list = [-1.4, 1.5, 1.9, -2.8]
    with pytest.raises(TypeError):
        string_to_float_list(string, float_list)


def test_string_to_float_list_too_many_inputs():
    string = "-1.4,1.5,1.9,-2.8"
    float_list = [-1.4, 1.5, 1.9, -2.8]
    assert float_list == string_to_float_list(string)


def test_string_to_float_list_no_float():
    string = "hello, world"
    with pytest.raises(ValueError):
        string_to_float_list(string)
```



```
test/test_pages_when_user_not_logged_in.py::test_capture_video_redirect PASSED                    [ 29%]
test/test_pages_when_user_not_logged_in.py::test_upload_file_no_redirect PASSED                   [ 30%]
test/test_pages_when_user_not_logged_in.py::test_capture_video_no_redirect PASSED                 [ 31%]
test/test_pages_when_user_not_logged_in.py::test_body PASSED                                      [ 33%]
test/test_pages_when_user_not_logged_in.py::test_uploaded PASSED                                  [ 34%]
test/test_user.py::test_new_user_with_fixture PASSED                                              [ 36%]
test/test_user.py::test_new_user PASSED                                                           [ 37%]
test/unittest/test_app_functions.py::test_string_to_float_list_all_positive PASSED                [ 38%]
test/unittest/test_app_functions.py::test_string_to_float_list_incorrect_type PASSED              [ 40%]
test/unittest/test_app_functions.py::test_string_to_float_list_all_negative PASSED                [ 41%]
test/unittest/test_app_functions.py::test_string_to_float_list_positive_negative PASSED           [ 43%]
test/unittest/test_app_functions.py::test_string_to_float_list_too_many_inputs PASSED             [ 44%]
test/unittest/test_app_functions.py::test_string_to_float_list_no_float PASSED                    [ 45%]
test/unittest/test_app_functions.py::test_float_list_to_string_all_positive PASSED                [ 47%]
test/unittest/test_app_functions.py::test_float_list_to_string_incorrect_type PASSED              [ 48%]
test/unittest/test_app_functions.py::test_float_list_to_string_all_negative PASSED                [ 50%]
test/unittest/test_app_functions.py::test_float_list_to_string_positive_negative PASSED           [ 51%]
test/unittest/test_app_functions.py::test_float_list_to_string_too_many_inputs PASSED             [ 52%]
test/unittest/test_app_functions.py::test_get_rpe PASSED                                          [ 54%]
test/unittest/test_app_functions.py::test_get_rpe_wrong_type PASSED                               [ 55%]
test/unittest/test_app_functions.py::test_get_rpe_too_many_inputs PASSED                          [ 56%]
test/unittest/test_app_functions.py::test_rpe_chart PASSED                                        [ 58%]
test/unittest/test_app_functions.py::test_rpe_chart_not_reversed PASSED                           [ 59%]
test/unittest/test_app_functions.py::test_rpe_chart_big_difference PASSED                         [ 61%]
test/unittest/test_app_functions.py::test_rpe_chart_small_difference PASSED                       [ 62%]
test/unittest/test_body_functions.py::test_getMidPoint_all_positive PASSED                        [ 63%]
test/unittest/test_body_functions.py::test_getMidPoint_not_enough_inputs PASSED                   [ 65%]
test/unittest/test_body_functions.py::test_getMidPoint_too_many_inputs PASSED                     [ 66%]
test/unittest/test_body_functions.py::test_getMidPoint_incorrect_type PASSED                      [ 68%]
test/unittest/test_body_functions.py::test_getMidPoint_all_negative PASSED                        [ 69%]
test/unittest/test_body_functions.py::test_getMidPoint_large_ints PASSED                          [ 70%]
test/unittest/test_body_functions.py::test_getMidPoint_small_ints PASSED                          [ 72%]
test/unittest/test_body_functions.py::test_getAngleC_all_positive PASSED                          [ 73%]
test/unittest/test_body_functions.py::test_getAngleC_all_negative PASSED                          [ 75%]
test/unittest/test_body_functions.py::test_getAngleC_not_enough_inputs PASSED                     [ 76%]
test/unittest/test_body_functions.py::test_getAngleC_too_many_inputs PASSED                       [ 77%]
test/unittest/test_body_functions.py::test_getAngleC_incorrect_type PASSED                        [ 79%]
test/unittest/test_body_functions.py::test_angleC_all_positive PASSED                             [ 80%]
test/unittest/test_body_functions.py::test_angleC_large_ints PASSED                               [ 81%]
test/unittest/test_body_functions.py::test_angleC_negative_ints PASSED                            [ 83%]
test/unittest/test_body_functions.py::test_angleC_not_enough_inputs PASSED                        [ 84%]
test/unittest/test_body_functions.py::test_angleC_too_many_inputs PASSED                          [ 86%]
test/unittest/test_body_functions.py::test_angleC_incorrect_type PASSED                           [ 87%]
test/unittest/test_body_functions.py::test_getDistance_all_positive PASSED                        [ 88%]
test/unittest/test_body_functions.py::test_getDistance_all_negative PASSED                        [ 90%]
test/unittest/test_body_functions.py::test_getDistance_positive_negative PASSED                   [ 91%]
test/unittest/test_body_functions.py::test_getDistance_large_ints PASSED                          [ 93%]
test/unittest/test_body_functions.py::test_getDistance_small_ints PASSED                          [ 94%]
test/unittest/test_body_functions.py::test_getDistance_large_small_ints PASSED                    [ 95%]
test/unittest/test_body_functions.py::test_getDistance_not_enough_inputs PASSED                   [ 97%]
test/unittest/test_body_functions.py::test_getDistance_too_many_inputs PASSED                     [ 98%]
test/unittest/test_body_functions.py::test_getDistance_incorrect_type PASSED                      [100%]

=================================================== warnings summary ===================================================
test/test_S3.py::test_creation
  /Users/jordanvoss/.pyenv/versions/3.7.3/lib/python3.7/site-packages/boto/plugin.py:40: DeprecationWarning: the imp module is deprecated in favour of importlib; see the module's documentation for alternative us
es
    import imp

test/test_pages_when_user_logged_in.py::test_login
  /Users/jordanvoss/.pyenv/versions/3.7.3/lib/python3.7/site-packages/sqlalchemy/util/langhelpers.py:254: SADeprecationWarning: The 'postgres' dialect name has been renamed to 'postgresql'
    loader = self.auto_fn(name)

-- Docs: https://docs.pytest.org/en/stable/warnings.html
=============================================== 72 passed, 2 warnings in 7.61s ===============================================
jordanvoss@Jordans-MBP:~/2021-ca400-vossj2-ndomban2/src|develop⚡ ⇒ ga .
```

1.2. Flask Page Tests

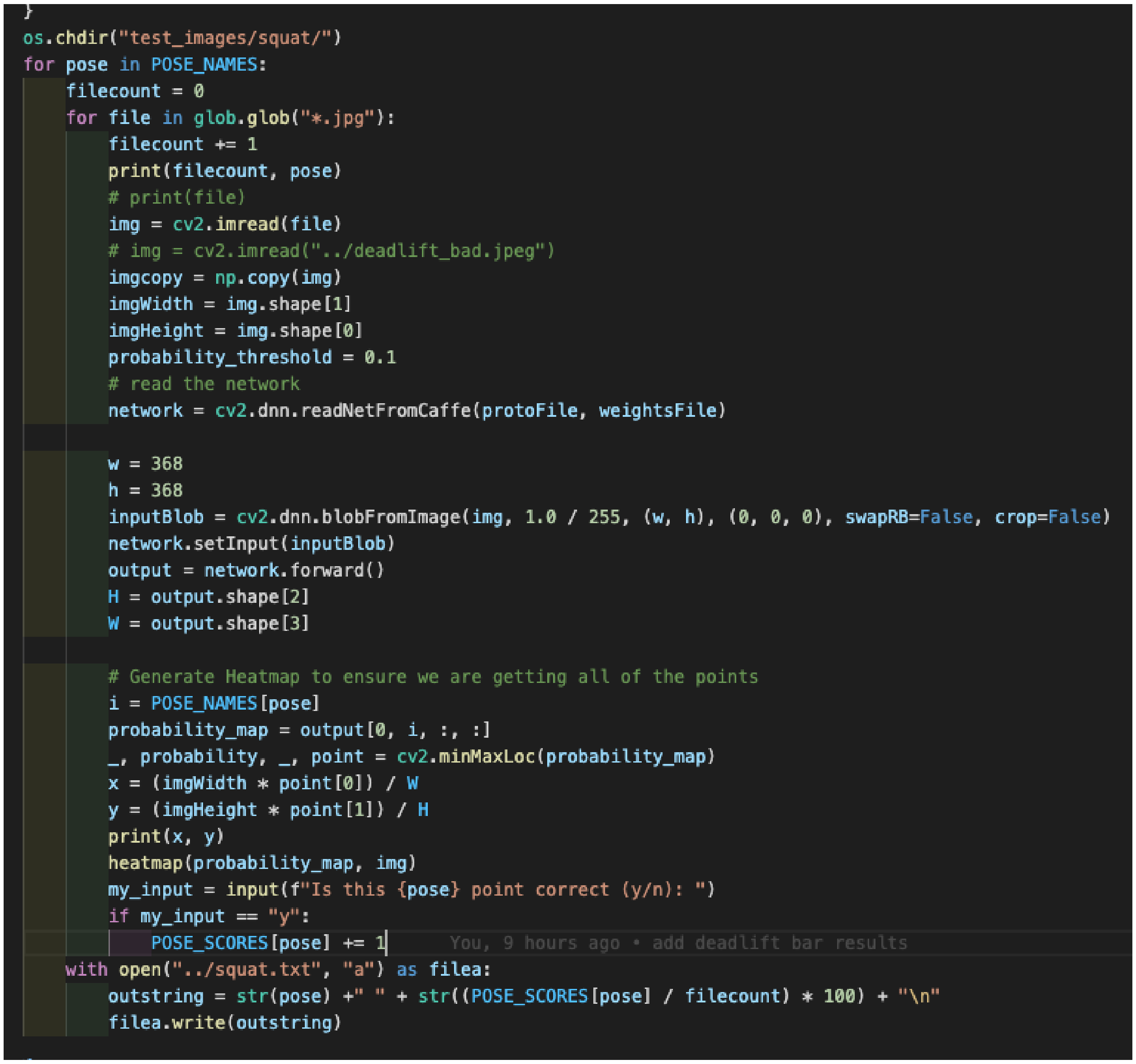
To test we get the correct HTTP code when we request each page, we also used pytest. Here we could assert the status code was correct when the user was either logged in or not to ensure security and that users couldn't navigate to any page they wanted without logging in prior to this.



```python
def test_home(app, client):
    res = client.get("/")
    assert res.status_code == 200
    assert b"<title>Homepage</title>" in res.data
    # client.assert_context("greeting", "hello")


def test_login(app, client):
    res = client.post("/login", data=dict(username="q", password="qqq"), follow_redirects=True)
    assert res.status_code == 200
    assert b"Login requested for user q, remember me= False" in res.data
    assert b"User q Logged in successfully !" in res.data


def test_video(app, client):
    res = client.post("/login", data=dict(username="q", password="qqq"), follow_redirects=True)
    assert res.status_code == 200
    assert b"Login requested for user q, remember me= False" in res.data
    assert b"User q Logged in successfully !" in res.data
    res = client.get("/video", follow_redirects=True)
    assert res.status_code == 200
    assert b"START RECORDING" in res.data
    assert b"STOP RECORDING" in res.data
```

2. Accuracy Tests

To test the accuracy of the body part tracking we used the PCK (Percentage of Correct Keypoints) metric to determine how accurate our tracking was compared to tracking without a barbell, as we know that the barbell blocking some body parts and the side on view also blocking more, that we may have less consistent body part accuracy. This is the test file we used. We ran it against frames from both squat and deadlift with and without the bar for 3 reps each.



```
}
os.chdir("test_images/squat/")
for pose in POSE_NAMES:
    filecount = 0
    for file in glob.glob("*.jpg"):
        filecount += 1
        print(filecount, pose)
        # print(file)
        img = cv2.imread(file)
        # img = cv2.imread("../deadlift_bad.jpeg")
        imgcopy = np.copy(img)
        imgWidth = img.shape[1]
        imgHeight = img.shape[0]
        probability_threshold = 0.1
        # read the network
        network = cv2.dnn.readNetFromCaffe(protoFile, weightsFile)

        w = 368
        h = 368
        inputBlob = cv2.dnn.blobFromImage(img, 1.0 / 255, (w, h), (0, 0, 0), swapRB=False, crop=False)
        network.setInput(inputBlob)
        output = network.forward()
        H = output.shape[2]
        W = output.shape[3]

        # Generate Heatmap to ensure we are getting all of the points
        i = POSE_NAMES[pose]
        probability_map = output[0, i, :, :]
        _, probability, _, point = cv2.minMaxLoc(probability_map)
        x = (imgWidth * point[0]) / W
        y = (imgHeight * point[1]) / H
        print(x, y)
        heatmap(probability_map, img)
        my_input = input(f"Is this {pose} point correct (y/n): ")
        if my_input == "y":
            POSE_SCORES[pose] += 1      You, 9 hours ago • add deadlift bar results
    with open("../squat.txt", "a") as filea:
        outstring = str(pose) +" " + str((POSE_SCORES[pose] / filecount) * 100) + "\n"
        filea.write(outstring)
```

These were our results:

Squat with bar PCK:

```
HEAD98.0
NECK79.0
RSHOULDER74.0
RELBOW51.0
RHAND44.0
```

```
LSHOULDER35.0
LELBOW11.0
LHAND36.0
RHIP56.99999999999999
RKNEE73.0
RANKLE86.0
LHIP64.0
LKNEE78.0
LANKLE75.0
CHEST88.0
TOTAL63.27
```

Squat without bar PCK:

```
HEAD 81.81818181818183
NECK 92.42424242424242
RSHOULDER 77.27272727272727
RELBOW 63.63636363636363
RHAND 56.060606060606055
LSHOULDER 62.121212121212125
LELBOW 25.757575757575758
LHAND 83.33333333333334
RHIP 59.09090909090909
RKNEE 84.84848484848484
RANKLE 96.96969696969697
LHIP 59.09090909090909
LKNEE 89.39393939393939
LANKLE 92.42424242424242
CHEST 78.78787878787878
TOTAL 73.535353535354
```

Deadlift with bar PCK:

```
HEAD98.38709677419355
NECK91.93548387096774
RSHOULDER83.87096774193549
RELBOW88.70967741935483
RHAND64.51612903225806
LSHOULDER91.93548387096774
LELBOW85.48387096774194
LHAND90.32258064516128
RHIP79.03225806451613
RKNEE83.87096774193549
RANKLE91.93548387096774
LHIP82.25806451612904
```

```
LKNEE80.64516129032258
LANKLE87.09677419354838
CHEST90.32258064516128
TOTAL86.021505376344
```

Deadlift without bar PCK:

```
HEAD 95.52238805970148
NECK 92.53731343283582
RSHOULDER 92.53731343283582
RELBOW 94.02985074626866
RHAND 82.08955223880598
LSHOULDER 91.04477611940298
LELBOW 97.01492537313433
LHAND 100.0
RHIP 86.56716417910447
RKNEE 100.0
RANKLE 97.01492537313433
LHIP 91.04477611940298
LKNEE 100.0
LANKLE 97.01492537313433
CHEST 94.02985074626866
TOTAL 94.029850746269
```

This gave us an average of 74.64% total accuracy when getting the body key points when using the bar, and 83.78% without it. We can see there's not a huge difference here. But definitely big enough to make a difference. There are times during the body tracking that it is evident that the tracking does not find the key point correctly all of the time. But it is still accurate enough to score and give information on a set.

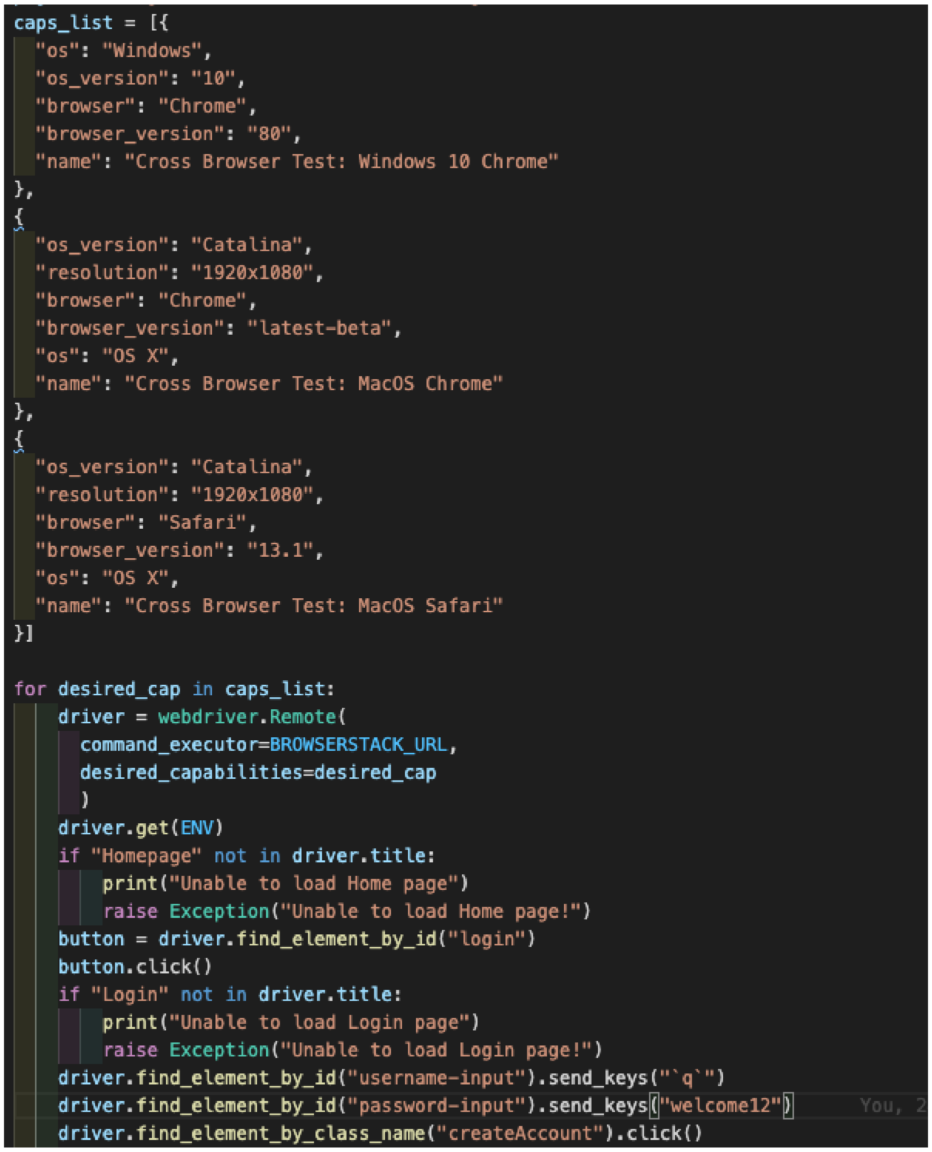
The average for both of the squats with and without the bar was 68.4% compared to the deadlift's 90%. This is along the lines of what we expected. As when the athlete is squatting their hands may be covering their shoulders, as well as their neck and/or chest. This increases when using the bar as you can see from the results. When we get to the bottom of the squat, the weight plates on the bar obstruct many body parts from view and cause even the highly visible body parts to the camera to be hidden. We only used a small weight plate for our videos, but when using larger ones this problem would increase. The deadlift on the other hand, the arms are straight and are easier to find. The plates are still an issue, mostly on the lower body and wrists, but with our check_points file for all lifts, we are able to predict a lot of these using the body's ratios as described in our Technical Manual.

The fast that the user is also facing to one side causes half of the body's key points to be obstructed. Luckily the system is able to recognise a lot of these, but it does cause some diminishing in the PCK values on the hidden side.

3. Behaviour Driver Development Tests

3.1. Cross Browser Testing

Our cross browser testing was implemented to make sure that the app would work successfully on all platforms, both windows and mac and across chrome and safari. This could be extended to iphone and android upon further development if we were to create native apps for these platforms and we are given a video depicting the journey that the system has tested. To implement this we used selenium web driver and there is an example of the code used to test it below.



```
caps_list = [{
  "os": "Windows",
  "os_version": "10",
  "browser": "Chrome",
  "browser_version": "80",
  "name": "Cross Browser Test: Windows 10 Chrome"
},
{
  "os_version": "Catalina",
  "resolution": "1920x1080",
  "browser": "Chrome",
  "browser_version": "latest-beta",
  "os": "OS X",
  "name": "Cross Browser Test: MacOS Chrome"
},
{
  "os_version": "Catalina",
  "resolution": "1920x1080",
  "browser": "Safari",
  "browser_version": "13.1",
  "os": "OS X",
  "name": "Cross Browser Test: MacOS Safari"
}]

for desired_cap in caps_list:
    driver = webdriver.Remote(
      command_executor=BROWSERSTACK_URL,
      desired_capabilities=desired_cap
      )
    driver.get(ENV)
    if "Homepage" not in driver.title:
        print("Unable to load Home page")
        raise Exception("Unable to load Home page!")
    button = driver.find_element_by_id("login")
    button.click()
    if "Login" not in driver.title:
        print("Unable to load Login page")
        raise Exception("Unable to load Login page!")
    driver.find_element_by_id("username-input").send_keys("`q`")
    driver.find_element_by_id("password-input").send_keys("welcome12")      You, 2
    driver.find_element_by_class_name("createAccount").click()
```

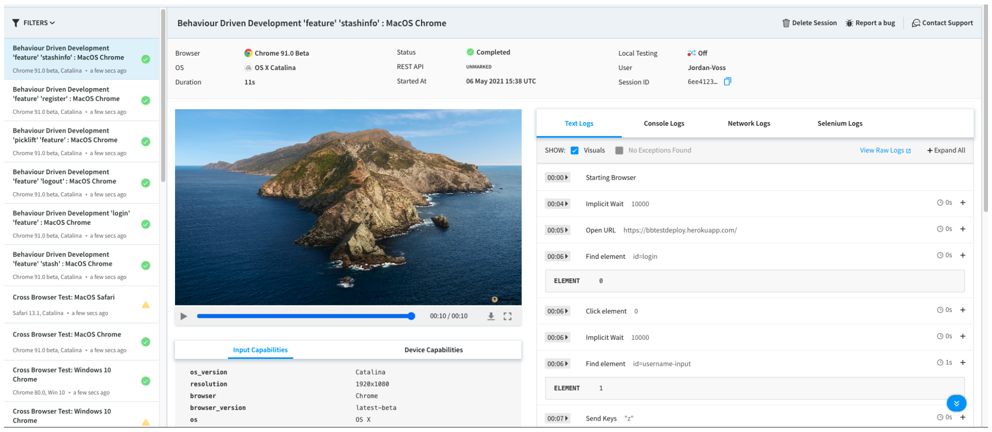
Here you can see the difference browsers and operating systems we used, as well as the selenium web driver instructions to follow the journey of navigating to the home page and logging in.

3.2. Gherkin Tests

We decided to use gherkin as our primary ui testing, as it allowed for an easier understanding for others looking at the tests that may not comprehend fully the technical jargon used when testing software. The behaviour driven development ideology uses a Given, When, Then setting for the scenario. I.e. Given a user, when the user logs in, then the logged in screen shows. We can implement functions for each of these strings, but the other person, for example a business analyst in a company who may not understand fully the testing jargon, can still see that the test failed where 'The logged in screen shows" so we know that there is an error on that part of the scenario. We implemented many scenarios which are run by our feature files as shown below. There is also the function implementation of some of the strings shown. These also gave us videos on the browserstack automate site. Which allowed us, when a test was failing, to see exactly what was happening on the test user's screen. This is particularly helpful when running a live product if a customer has a problem it is easy to recreate the journey to get to the problem and once the tests pass the issue has been resolved.

This link will show you the example of the stash info video that was generated from one of our tests: https://www.youtube.com/watch?v=bzK3PMJrT8E

Below is the dashboard on browserstack automate where we can see any of the videos that have been generated from the many tests that we have written, as well as the test results in the command line that came from running them.



Behaviour Driven Development 'feature' 'stashinfo' : MacOS Chrome — Chrome 91.0 beta, Catalina • a few secs ago
Behaviour Driven Development 'feature' 'register' : MacOS Chrome — Chrome 91.0 beta, Catalina • a few secs ago
Behaviour Driven Development 'picklift' 'feature' : MacOS Chrome — Chrome 91.0 beta, Catalina • a few secs ago
Behaviour Driven Development 'feature' 'logout' : MacOS Chrome — Chrome 91.0 beta, Catalina • a few secs ago
Behaviour Driven Development 'login' 'feature' : MacOS Chrome — Chrome 91.0 beta, Catalina • a few secs ago
Behaviour Driven Development 'feature' 'stash' : MacOS Chrome — Chrome 91.0 beta, Catalina • a few secs ago
Cross Browser Test: MacOS Safari — Safari 13.1, Catalina • a few secs ago
Cross Browser Test: MacOS Chrome — Chrome 91.0 beta, Catalina • a few secs ago
Cross Browser Test: Windows 10 Chrome — Chrome 80.0, Win 10 • a few secs ago
Cross Browser Test: Windows 10 Chrome

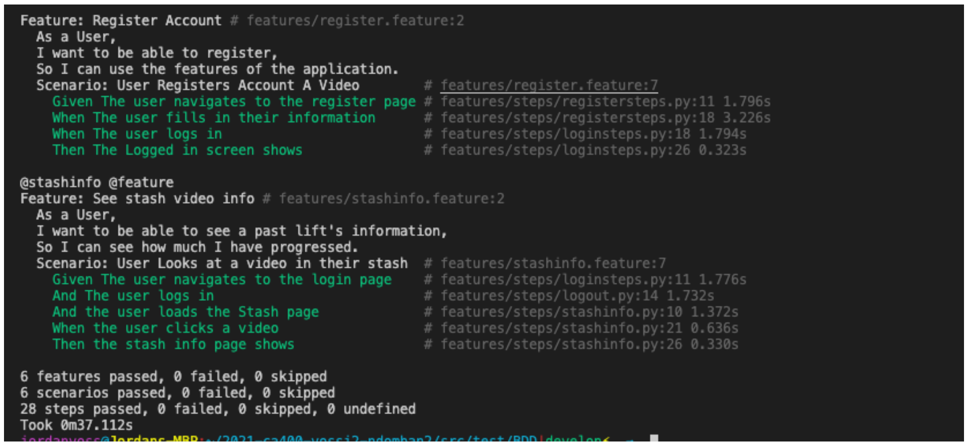
Behaviour Driven Development 'feature' 'stashinfo' : MacOS Chrome

Browser	Chrome 91.0 Beta	Status	Completed	Local Testing	Off
OS	OS X Catalina	REST API	UNMARKED	User	Jordan-Voss
Duration	11s	Started At	06 May 2021 15:38 UTC	Session ID	6ee4123...

SHOW: ✓ Visuals ▨ No Exceptions Found

```
00:00  Starting Browser
00:04  Implicit Wait    10000                               0s
00:05  Open URL    https://bbtestdeploy.herokuapp.com/      0s
00:06  Find element    id=login                             0s
       ELEMENT    0
00:06  Click element    0                                   0s
00:06  Implicit Wait    10000                               0s
00:06  Find element    id=username-input                    1s
       ELEMENT    1
00:07  Send Keys    "z"                                     0s
```

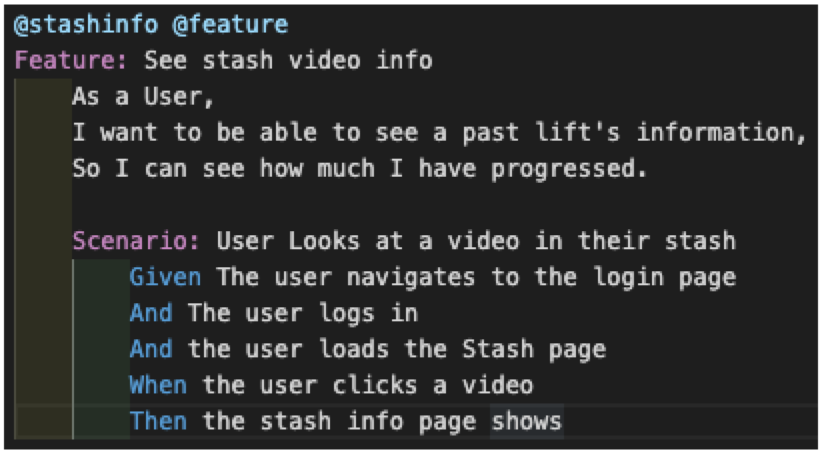
```
os_version        Catalina
resolution        1920x1080
browser           Chrome
browser_version   latest-beta
os                OS X
```



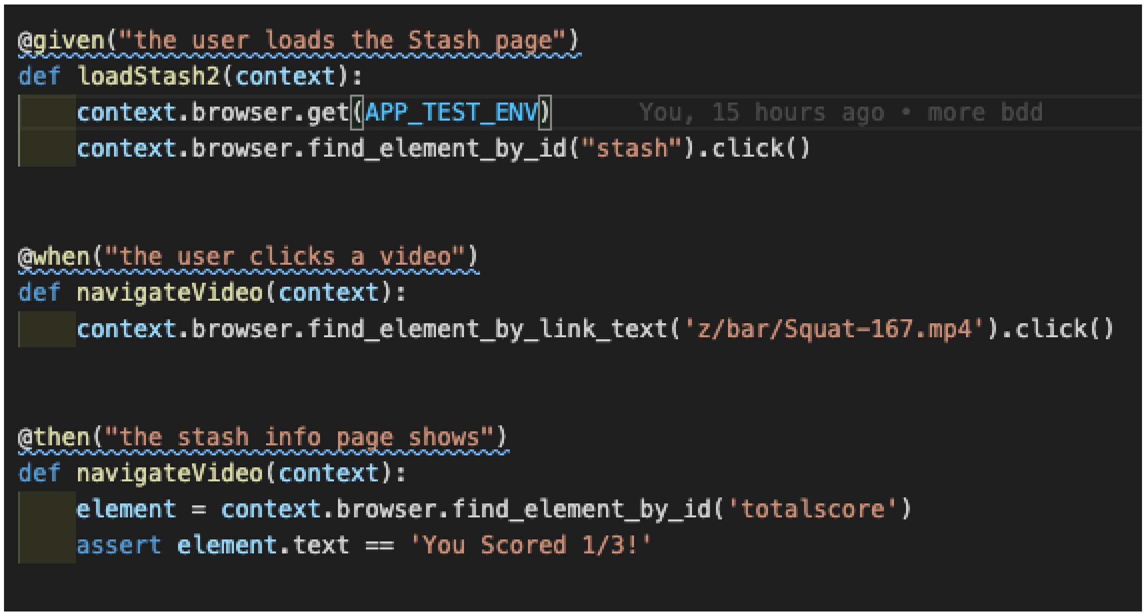
```
Feature: Register Account # features/register.feature:2
  As a User,
  I want to be able to register,
  So I can use the features of the application.
  Scenario: User Registers Account A Video          # features/register.feature:7
    Given The user navigates to the register page # features/steps/registersteps.py:11 1.796s
    When The user fills in their information      # features/steps/registersteps.py:18 3.226s
    When The user logs in                         # features/steps/loginsteps.py:18 1.794s
    Then The Logged in screen shows               # features/steps/loginsteps.py:26 0.323s

@stashinfo @feature
Feature: See stash video info # features/stashinfo.feature:2
  As a User,
  I want to be able to see a past lift's information,
  So I can see how much I have progressed.
  Scenario: User Looks at a video in their stash  # features/stashinfo.feature:7
    Given The user navigates to the login page    # features/steps/loginsteps.py:11 1.776s
    And The user logs in                          # features/steps/logout.py:14 1.732s
    And the user loads the Stash page             # features/steps/stashinfo.py:10 1.372s
    When the user clicks a video                  # features/steps/stashinfo.py:21 0.636s
    Then the stash info page shows                # features/steps/stashinfo.py:26 0.330s

6 features passed, 0 failed, 0 skipped
6 scenarios passed, 0 failed, 0 skipped
28 steps passed, 0 failed, 0 skipped, 0 undefined
Took 0m37.112s
```

Below you can see an example of one of the Gherkin feature files used to run the tests as well as the functions that correlate to the string in the feature file. It is evident how easy it would be for a 'non-tech' person to understand what is happening here and where the problems are arising.

These also help us to ensure that a new feature added doesn't break any currently working functionality of the app and using the videos we can see how the UI looks and make sure that everything is in the right place.



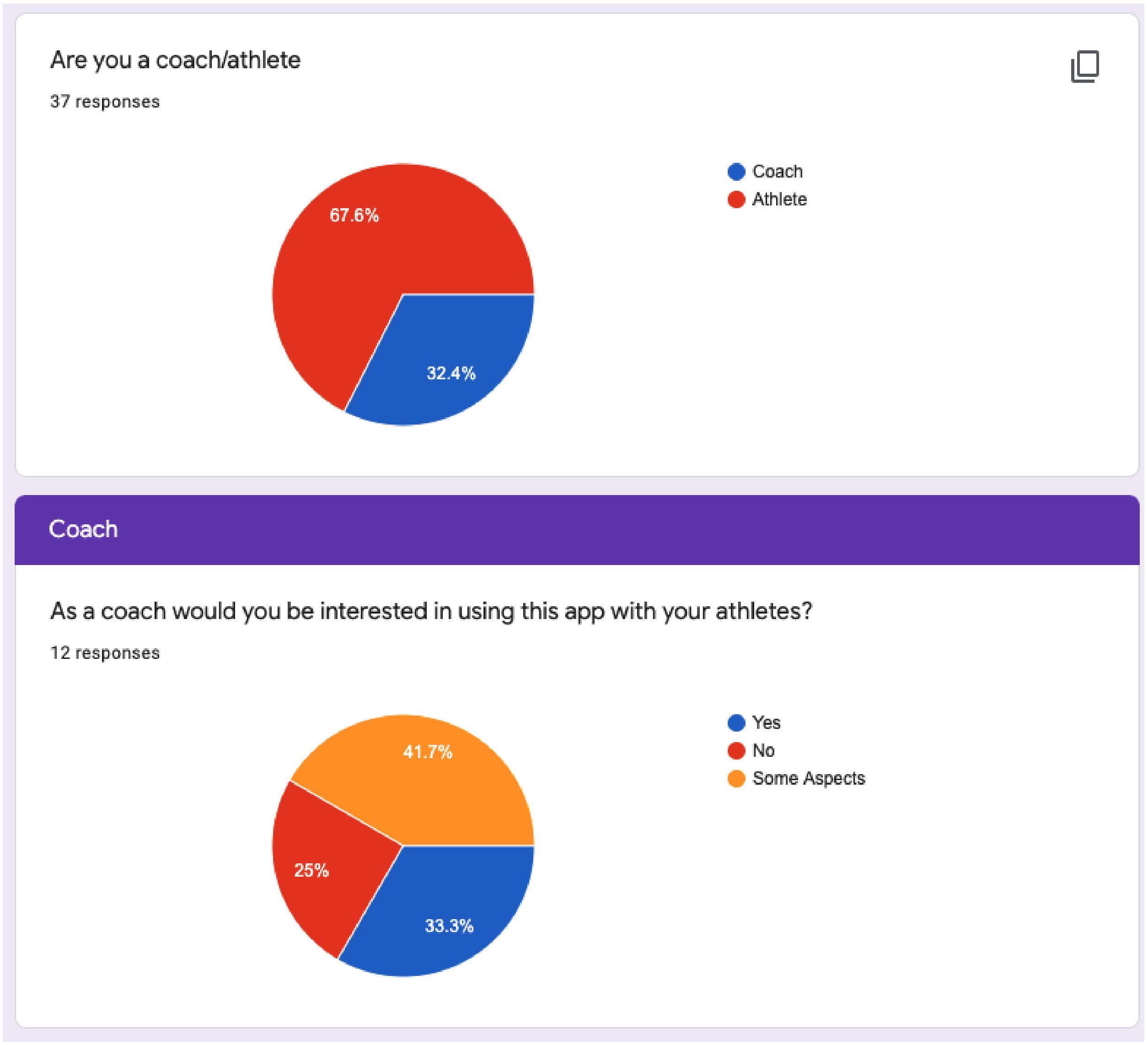
```
@stashinfo @feature
Feature: See stash video info
    As a User,
    I want to be able to see a past lift's information,
    So I can see how much I have progressed.

    Scenario: User Looks at a video in their stash
        Given The user navigates to the login page
        And The user logs in
        And the user loads the Stash page
        When the user clicks a video
        Then the stash info page shows
```



```
@given("the user loads the Stash page")
def loadStash2(context):
    context.browser.get(APP_TEST_ENV)        You, 15 hours ago • more bdd
    context.browser.find_element_by_id("stash").click()


@when("the user clicks a video")
def navigateVideo(context):
    context.browser.find_element_by_link_text('z/bar/Squat-167.mp4').click()


@then("the stash info page shows")
def navigateVideo(context):
    element = context.browser.find_element_by_id('totalscore')
    assert element.text == 'You Scored 1/3!'
```

4. User Tests

As part of our user testing we got over 30 people to use the app who were coaches, clients of coaches, or regular gym goers. As you can see the vast majority were athletes, but we had a few coaches to get the point of view of someone who works in this industry. As the app size was too large for heroku to support, we had to carry out manual testing, where the user could do everything on the app apart from recording or uploading a video. We got them to send us the videos and showed them the process when running on local host as they controlled the movements on the screen.


Are you a coach/athlete

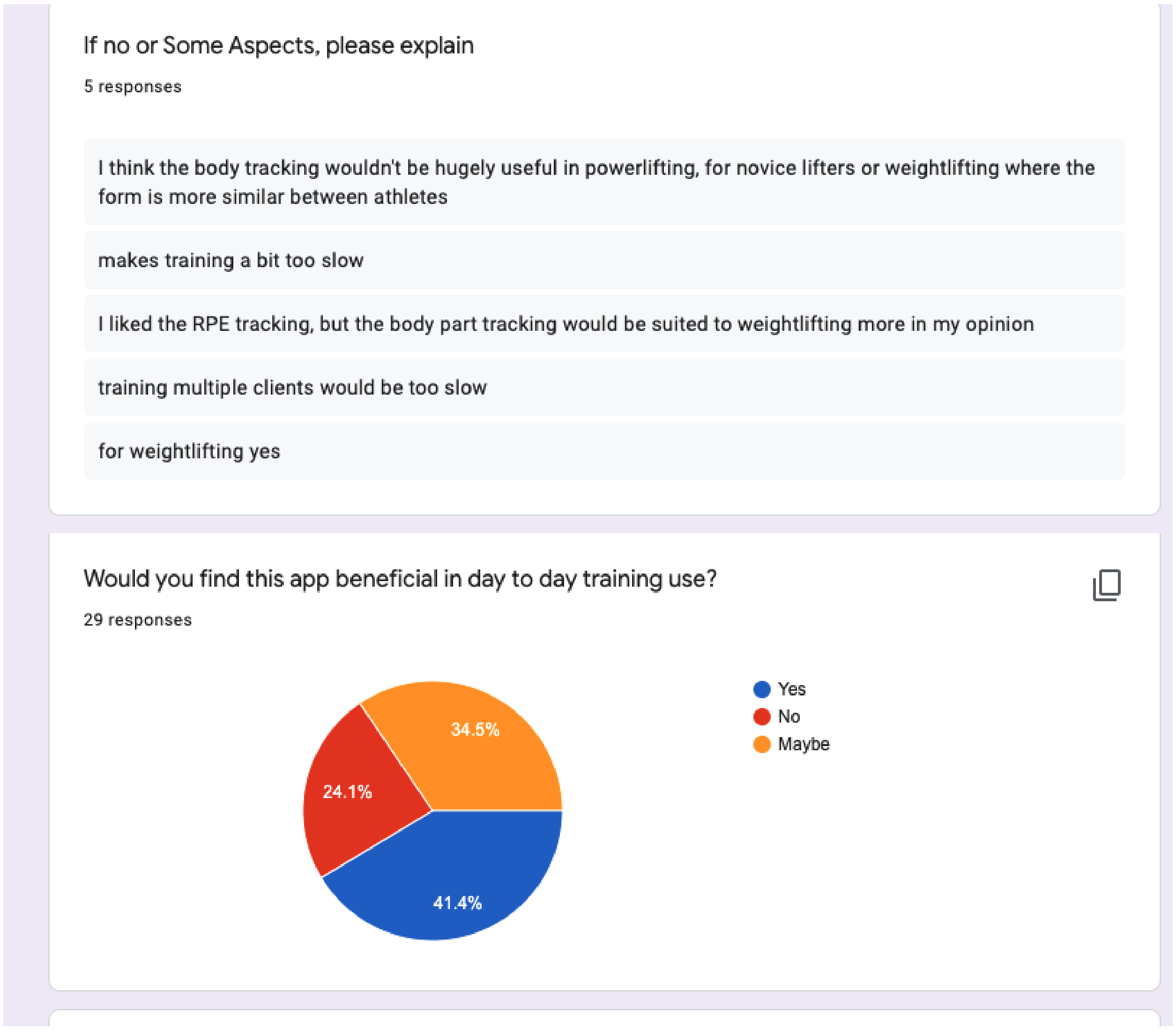
37 responses

Coach

As a coach would you be interested in using this app with your athletes?

12 responses

We can see here that some of the coaches said that they wouldn't like to use the app with their athletes, upon talking further with some of the coaches and from their more detailed responses below, we found that some coaches found the body tracking to be not quite as accurate or relevant as we had hoped. They said that the form in powerlifting would be very different for everyone, and if the app could critique form based on learned posture and body part positioning, then it could become a viable option. Most of the coaches agreed that the RPE section was useful and would be beneficial to them when introducing new clients to RPE as well as taking away some of the subjectivity of RPE for novice to advanced lifters.



If no or Some Aspects, please explain

5 responses

I think the body tracking wouldn't be hugely useful in powerlifting, for novice lifters or weightlifting where the form is more similar between athletes

makes training a bit too slow

I liked the RPE tracking, but the body part tracking would be suited to weightlifting more in my opinion

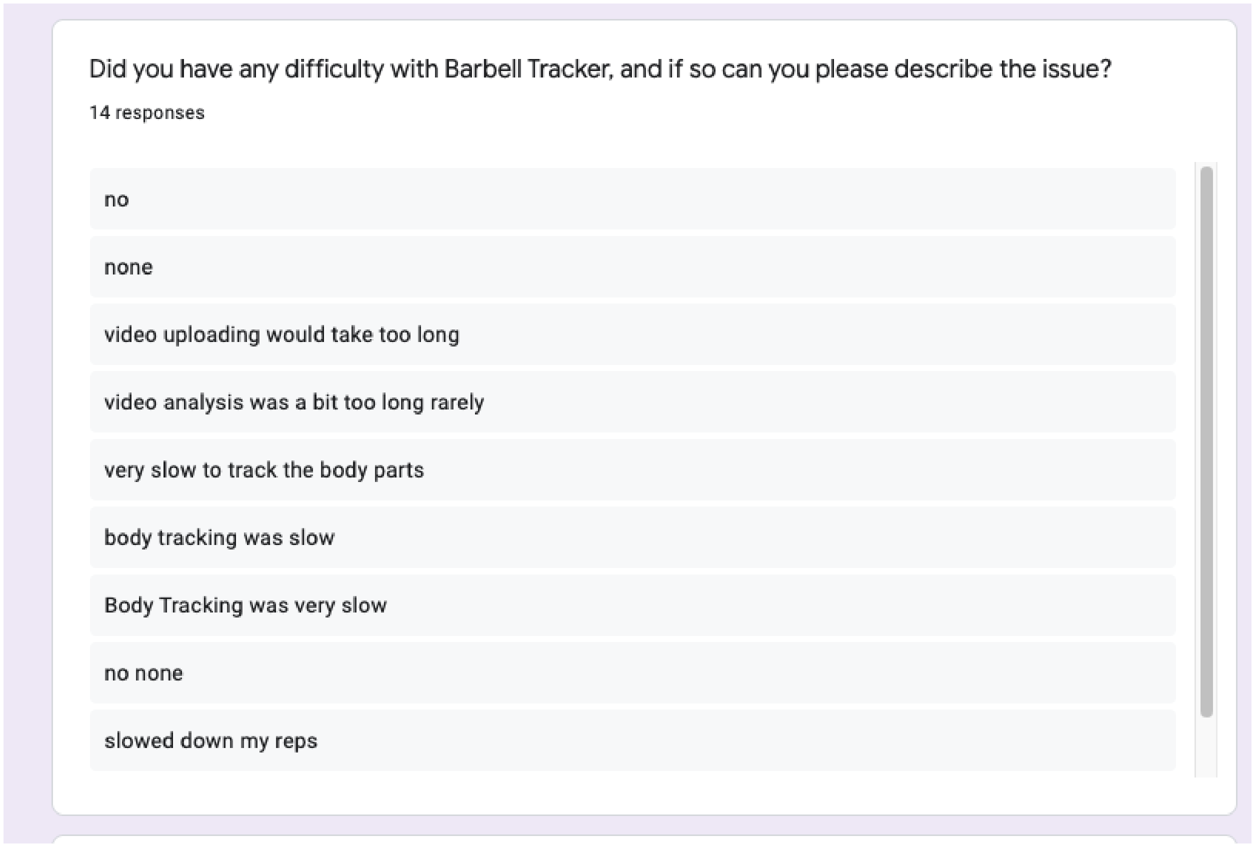
training multiple clients would be too slow

for weightlifting yes

Would you find this app beneficial in day to day training use?

29 responses

We can see from the above pie chart, that most people would find at least one portion of the application to be useful in their day to day training use. Upon further conversation with the users, we found that a significant portion of the lifters who said no, felt that it was too slow to be used day to day. As we can also see from the responses below.



Did you have any difficulty with Barbell Tracker, and if so can you please describe the issue?

14 responses

- no
- none
- video uploading would take too long
- video analysis was a bit too long rarely
- very slow to track the body parts
- body tracking was slow
- Body Tracking was very slow
- no none
- slowed down my reps

Overall from the user testing we got invaluable information with regards to the future work that could be implemented on our app. From Speeding up the tracking, to creating custom form checkers with personal posture references we could definitely further this app to become viable in the industry. You can see more about future implementations in the Technical Manual.