IT314 PROJECT 2020

COUPONS UTILIZATION AND REMINDER SYSTEM

TESTING

GROUP 7

DHIRUBHAI AMBANI INSTITUTE OF INFORMATION AND TECHNOLOGY

COUPONS UTILIZATION AND REMINDER SYSTEM

TESTING INFORMATION FOR AN APP

Group: 07

Team Members:

- *Dhandhalya Sagar - 201701139*
- *Patel Avi - 201701147*
- *Jarsaniya Tirth - 201701140*
- *Shah Meet - 201701143*
- *Thakkar Chintan - 201701141*
- *Hariyali Gajera - 201701119*
- *Ponnam Sai - 201701125*
- *Parmar Kaushal - 201701092*
- *Abhishek Jishu - 201701086*
- *Mohul Srivastva - 201701130*
- *Parikh Yash - 201701099*
- *Chauhan Vikas – 201701093*

TABLE OF CONTENTS

1. OVERVIEW

Coupon Assistant is meant to provide service which will help users to utilize available deals, offers, discounts and coupons which they have unlocked personally, or which are available in the market for all in a more appropriate, accurate, easy and comfortable way. This document will show how we have carried out different testing for our application. We tried to test each and every component of the application such that the density of bug (or error) is minimized. We know that exhausting testing is not possible as the number of different combinations for all inputs is very large. So, we applied some testing techniques to get assurance of minimized residual error density in our application.

2. DIFFERENT TYPES OF TESTING WE HAVE DONE

We used the ‘Black Box Testing’ as a method to test the specifications or requirements. There are two types of test selection technique in Black Box Testing.

1. Equivalence Class Partitioning
2. Boundary Value Analysis.

Since our App is implemented in a flutter software development kit, it provides three types of testing techniques to test the code.

1. Unit Testing
2. Widget Testing
3. Integration Testing

Since flutter is still in progress mode, there is no any testing tool available to execute white box testing (to test code). So, it was a good experience with testing flutter apps without using some additional tools and writing testing additional scripts to check different parts of source code of the application.

2.1 BLACK BOX TESTING

Tests generated by black box testing techniques are as follows:

Since we have implemented login features only using google sign in. There won’t be any valid or invalid partitioning. Following scenarios may happen while logging in the application.

1. User has signed out from google account then android system will automatically ask for sign in to google account.
2. If a user has multiple google accounts, then the system will ask to choose anyone account.

TEST FOR FEEDBACK SECTION

Constraint: User should be logged in. to submit the feedback, the user should select a number of stars. Additional comments text box can be empty.

TEST CASES	VALID / INVALID
Clicking on the feedback button without login to app	Invalid
Clicking on the feedback button after login.	Valid
Submitting feedback without selecting the number of stars and empty additional comments.	Invalid
Submitting feedback without selecting the number of stars and non-empty additional comments.	Invalid
Submitting feedback after selecting the number of stars with empty additional comments.	Valid
Submitting feedback after selecting the number of stars with non-empty additional comments	Valid

TEST FOR SETTING NOTIFICATION FOR A BOTH GENERAL AND PERSONAL COUPONS

Constraint: Selected date should be between current date and expiry date of that particular coupon (inclusively).

Let’s selected date is x, current date is y and expiry date is z.

TEST CASES	VALID/INVALID
x=y	Valid

x=z	Valid
x>y && x<z	Valid
x<y	Invalid
x>z	Invalid

TEST FOR ‘ADD COUPON’

Input: Company name, Coupon code, Expiry date, Discount, T&C, other details.

Constraint: Users should select any one company from a dropdown list (that means non-empty). Coupon code should be non-empty. Expiry date should be any future date. Discount is integer number and it should be between 1 and 100 (inclusively). T&C and other details information can be empty or non-empty.

Let’s denote company name as CN, coupon code as CC, expiry date as ED, discount as DC, other details as OD. Since T&C and OD can be anything empty or non-empty, let’s not include them in test cases.

TEST CASES	**VALID/ INVALID**
Non-empty CN, non-empty CC, ED is date of tomorrow, DC = 1	Valid
Non-empty CN, non-empty CC, ED is date of tomorrow, DC = 100	Valid
Non-empty CN, non-empty CC, ED is date of tomorrow, DC = 50	Valid

Non-empty CN, non-empty CC, ED is any future date other than date of tomorrow, DC = 1	Valid
Non-empty CN, non-empty CC, ED is any future date other than date of tomorrow, DC = 100	Valid
Non-empty CN, non-empty CC, ED is any future date other than date of tomorrow, DC = 50	Valid
If CN is empty or CC is empty, or ED is any past date or DC is not in range [1,100]	Invalid

TEST FOR 'ADD COUPON WITH IMAGE'

Input: User is provided with two options. One is 'Pick an image' and the other is 'Click an image'. In former option, user will pick an image from phone storage or from any drive. In later option user will click an image using mobile camera.

Constraint: User should pick an image that contains relevant information for coupon to be added. Mobile camera should be good enough to capture the image with sufficient quality.

TEST CASES	**VALID/ INVALID**
Selected Option is 'Pick an image' and user has selected image containing relevant info.	Valid
Selected Option is 'Pick an image' and user has selected image containing irrelevant info.	Invalid

Selected Option is ‘Pick an image’ and user has not selected any image.	Invalid
Selected Option is ‘Click an image’ and user has clicked image containing relevant info.	Valid
Selected Option is Click an image’ and the user has clicked an image containing irrelevant info.	Invalid
Selected Option is Click an image’ and the user has not clicked any image.	Invalid

2.2 UNIT TESTING

Unit tests are handy for verifying the behavior of a single function, method, or class. The ‘test’ package provides the core framework for writing unit tests. This package will help you to run logic or a single function we can compare the expected and actual results. Now let’s take some examples from our app unit testing.

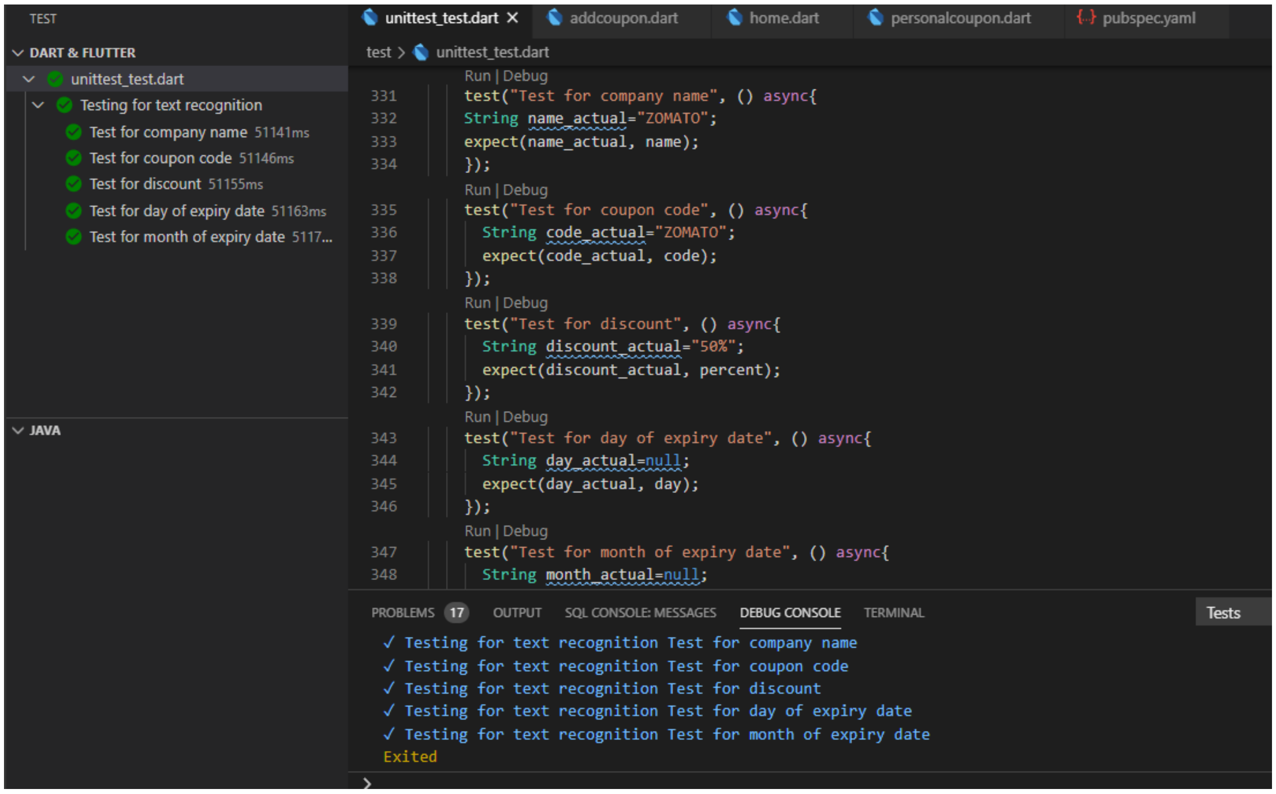
1. Delete coupon function test:

   This function will delete all coupons from the database which have expiry date less than today’s date. To test this function, we mock firestore and add an expired coupon to that database. After that run this function, after running the function the number of coupons is the same as before then the function is working correctly. If not, then the test will fail.

2. Detecting coupon details from text which is recognized from image of coupon:

   This function detects the coupon details from text given to it. Text is generated through image processing of image which contains coupon details. To test this function, we first ran our app in debug mode, gave an image of the coupon and the text generated by image processing is stored locally. In the test function we called the function which detects the coupon details and then we compared the actual result with the expected result. Below image shows the result for the input text: 00:06 G **zomato** When is the right time to eat? Always! Hey, foodie! We are all set to delight you with an mmmazing offer.Use **code ZOMATO** and enjoy up to **50%** off on your next order. Pick your favourites from our wide range

```
of cuisines and satisfy all your cravings now! TCA. Begin the binge For
the love of food, Team Zomato In case you wish to stop receiving emails
from Zomato, please unsubscribe here O.
```


TEST
DART & FLUTTER
unittest_test.dart
Testing for text recognition
Test for company name 51141ms
Test for coupon code 51146ms
Test for discount 51155ms
Test for day of expiry date 51163ms
Test for month of expiry date 5117...
JAVA
unittest_test.dart
addcoupon.dart
home.dart
personalcoupon.dart
pubspec.yaml
test > unittest_test.dart
PROBLEMS 17
OUTPUT
SQL CONSOLE: MESSAGES
DEBUG CONSOLE
TERMINAL
Tests

```
Run | Debug
test("Test for company name", () async{
String name_actual="ZOMATO";
expect(name_actual, name);
});
Run | Debug
test("Test for coupon code", () async{
  String code_actual="ZOMATO";
  expect(code_actual, code);
});
Run | Debug
test("Test for discount", () async{
  String discount_actual="50%";
  expect(discount_actual, percent);
});
Run | Debug
test("Test for day of expiry date", () async{
  String day_actual=null;
  expect(day_actual, day);
});
Run | Debug
test("Test for month of expiry date", () async{
  String month_actual=null;
```

```
✓ Testing for text recognition Test for company name
✓ Testing for text recognition Test for coupon code
✓ Testing for text recognition Test for discount
✓ Testing for text recognition Test for day of expiry date
✓ Testing for text recognition Test for month of expiry date
Exited
```

2.3 WIDGET TESTING

First i would like to say everything is widget in flutter, as like everything is Activity in android. So, we can divide whole app UI into the widget, as like home page of the app will show by home widget. So, by widget test we can test specific parts of UI and their functionality. you need a few additional tools provided by the '**flutter test**' package, which ships with the Flutter SDK with help of this package we render a specific widget and in test environment , and check for specific button, icon and other component is available in UI, and check their functionality, Examples from our app.

1. Add new coupon Form widget test:
   a. There is a form in app which will take coupon details from user and add it to the database. Total 5 text box is available here, so with widget test we can check for all textbox are visible and we can programmatically enter text into the text box, we can check if input is not valid than UI give error. These are some cases that we implemented.

i. If coupon code text field is null, then give error
ii. If discount > 100 then check for the error.

2. Submit feedback widget test:
   a. There is an option to submit feedback which takes input as a number of stars and additional comments. There are three things in that form, one is a text box, second is a smooth star rating bar and third is a submit button. Text box optional but star rating is mandatory. We programmatically enter text into the text. So, if the test submit button without selecting star rating, the test will give an error. And if we test the submit button after filling necessary details it will pass the test.

Above examples is given for the two widgets in our app we have main 8-9 widget(pages) all widget is tested with the widget test. So, if you change your UI then you can run a widget test to test to check these changes are not affecting other functions of this widget.

2.4 INTEGRATION TESTING

Unit tests and widget tests are handy for testing individual classes, functions, or widgets. However, they generally don't test how individual pieces work together as a whole or capture the performance of an application running on a real device. These tasks are performed with ***integration tests***. Integration tests work as a pair: first, deploy an instrumented application to a real device or emulator and then "drive" the application from a separate test suite, checking to make sure everything is correct along the way.

Basically, with a flutter driver we automate the app, like users do tap, and drag. We can do all this thing by driver programmed, like selenium web driver automate web app. So, integration test will start from the beginning install app on the device, try clicking button and enter text in field as instructed.

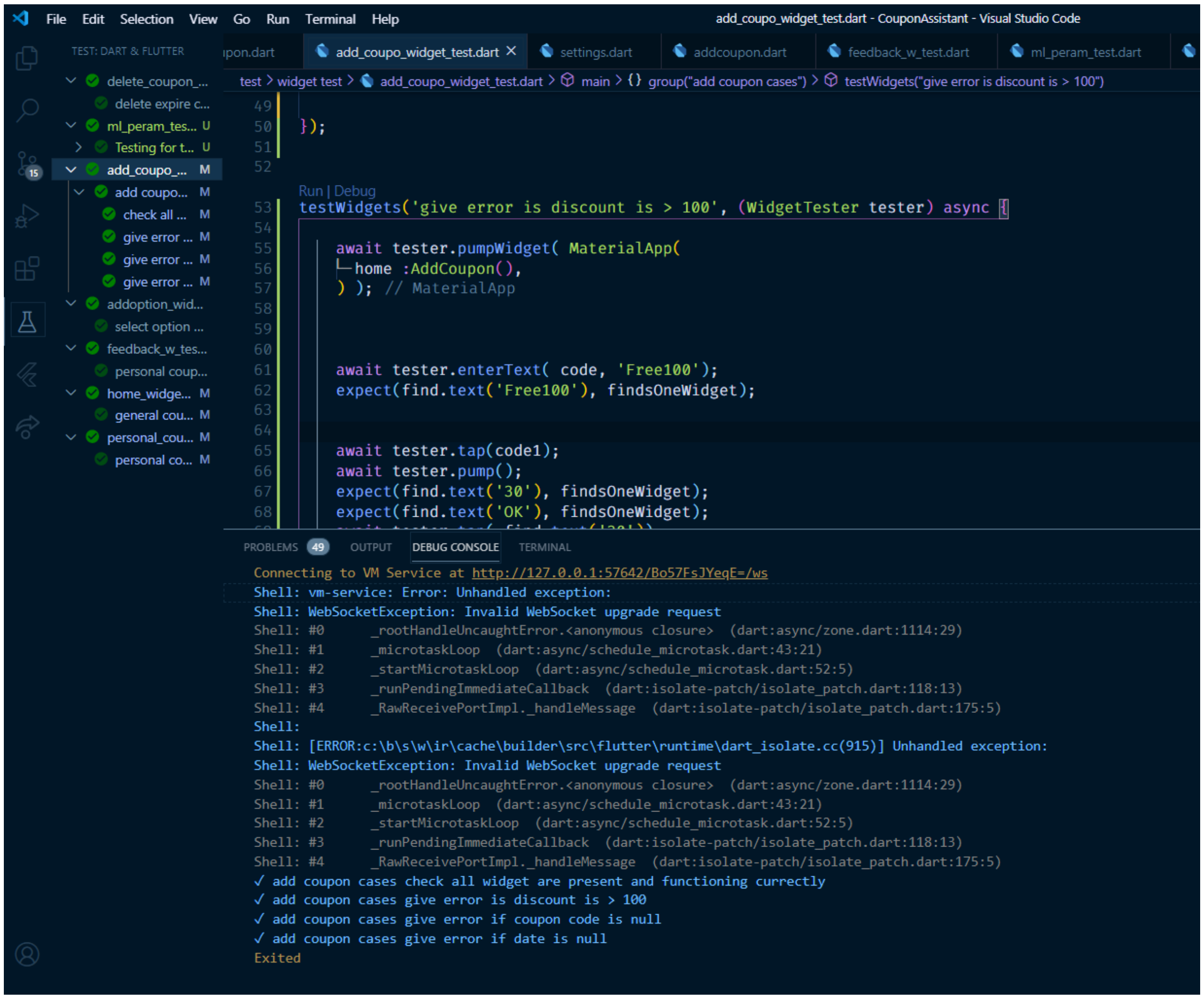
In addition to that you can see performance in real time like which page is taking a long time to load and how much RAM apps are using and many other important information. And you can take screenshots when the test is running. to check what caused the problem.

One of the limitations of the flutter driver is you cannot automate login. Video of the integration testing is here.

All test source code for the unit test and widget test are here.

And Integration testing is here.

Flutter does not provide test report like JUnit do so to prove our all test are passed I would like to include a screenshot.

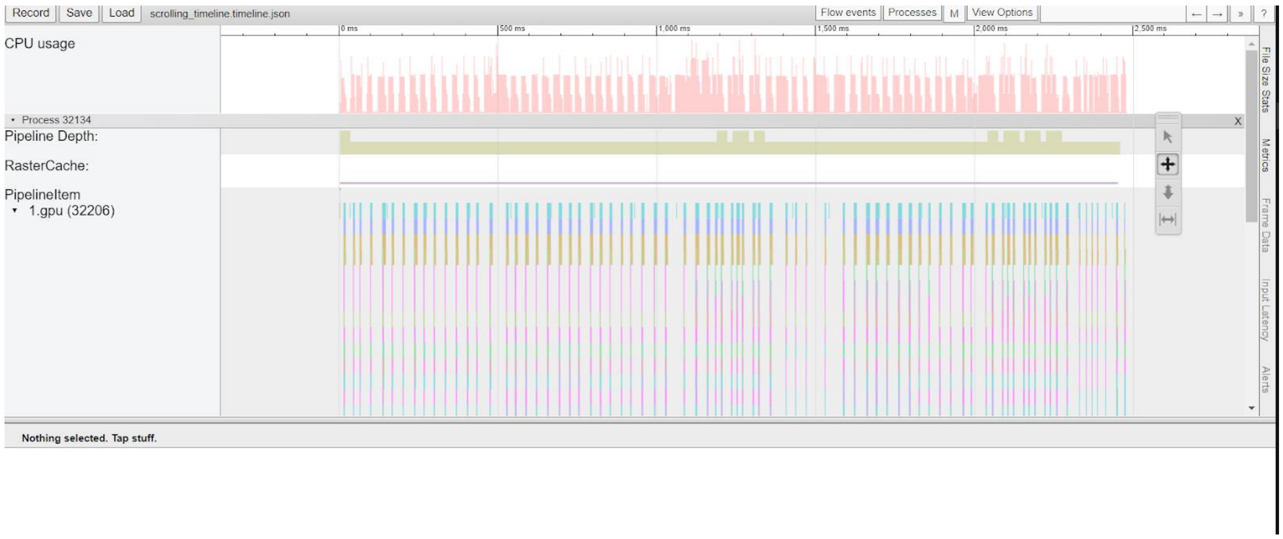


```
});

testWidgets('give error is discount is > 100', (WidgetTester tester) async {

    await tester.pumpWidget( MaterialApp(
    home :AddCoupon(),
    ) ); // MaterialApp


    await tester.enterText( code, 'Free100');
    expect(find.text('Free100'), findsOneWidget);


    await tester.tap(code1);
    await tester.pump();
    expect(find.text('30'), findsOneWidget);
    expect(find.text('OK'), findsOneWidget);
```

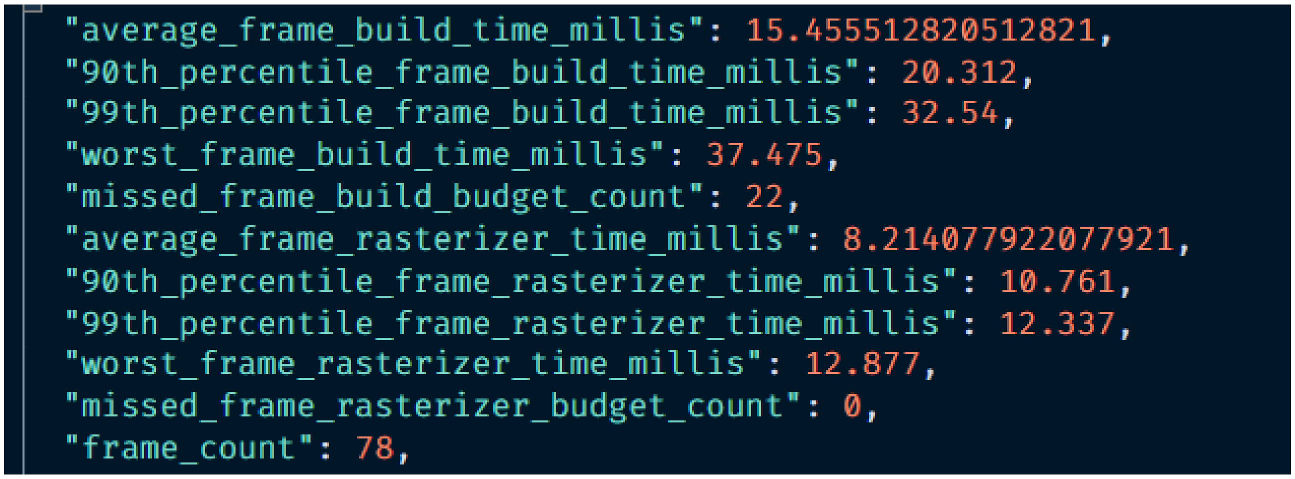
```
Connecting to VM Service at http://127.0.0.1:57642/Bo57FsJYeqE=/ws
Shell: vm-service: Error: Unhandled exception:
Shell: WebSocketException: Invalid WebSocket upgrade request
Shell: #0      _rootHandleUncaughtError.<anonymous closure>  (dart:async/zone.dart:1114:29)
Shell: #1      _microtaskLoop  (dart:async/schedule_microtask.dart:43:21)
Shell: #2      _startMicrotaskLoop  (dart:async/schedule_microtask.dart:52:5)
Shell: #3      _runPendingImmediateCallback  (dart:isolate-patch/isolate_patch.dart:118:13)
Shell: #4      _RawReceivePortImpl._handleMessage  (dart:isolate-patch/isolate_patch.dart:175:5)
Shell:
Shell: [ERROR:c:\b\s\w\ir\cache\builder\src\flutter\runtime\dart_isolate.cc(915)] Unhandled exception:
Shell: WebSocketException: Invalid WebSocket upgrade request
Shell: #0      _rootHandleUncaughtError.<anonymous closure>  (dart:async/zone.dart:1114:29)
Shell: #1      _microtaskLoop  (dart:async/schedule_microtask.dart:43:21)
Shell: #2      _startMicrotaskLoop  (dart:async/schedule_microtask.dart:52:5)
Shell: #3      _runPendingImmediateCallback  (dart:isolate-patch/isolate_patch.dart:118:13)
Shell: #4      _RawReceivePortImpl._handleMessage  (dart:isolate-patch/isolate_patch.dart:175:5)
✓ add coupon cases check all widget are present and functioning currectly
✓ add coupon cases give error is discount is > 100
✓ add coupon cases give error if coupon code is null
✓ add coupon cases give error if date is null
Exited
```

2.5 PERFORMANCE TESTING

Flutter also provides some performance testing tool so we try that, by using this tool you can see CPU and other resources used by your app and you can customize it accordingly.



Test summary of driver test,



```
  "average_frame_build_time_millis": 15.455512820512821,
  "90th_percentile_frame_build_time_millis": 20.312,
  "99th_percentile_frame_build_time_millis": 32.54,
  "worst_frame_build_time_millis": 37.475,
  "missed_frame_build_budget_count": 22,
  "average_frame_rasterizer_time_millis": 8.214077922077921,
  "90th_percentile_frame_rasterizer_time_millis": 10.761,
  "99th_percentile_frame_rasterizer_time_millis": 12.337,
  "worst_frame_rasterizer_time_millis": 12.877,
  "missed_frame_rasterizer_budget_count": 0,
  "frame_count": 78,
```