Detailed Description

Methods and examples

1. Method: One action and one result

Example 1. (One action - One result)

```
Scenario Name: < User with administrator role is to be logged in >
Expected result: < User is on ~/Administration  page >
```

Example 2.(One action - One result)
Scenario 1

```
Scenario Name: < Click on [Join] button on videoplayer triggers redirection on "Join" page >
Expected result: < "Join" page is loaded >
```

Automated test preview

```

@Test(priority = 111)
public void joinNowButtonOnVideoPlayerPageRedirectsToJoinPageTest() {
def url = "f1-racing/video/abu-dhabi/88"
videoPageHelper.navigateUnauthorized(url)
videoPageHelper.clickJoinNowButton()
assertTrue(joinPageHelper.isPageLoaded(), "'Join Now' Button is not redirect to Join Page")
}
```

2. Method: One action - multiple results

Example 1. (One action - One multiple results)

```
Scenario Name: < User with administrator role is to be logged in >
Expected results:
< User is on ~/Administration  page >
< User has available link for his profile page (~/Profile) >
< User UI contains available [LogOut] action >
```

Example 2.(One action - multiple results)
Scenario 1. One action and several

```

Scenario Name: < User is able to upload 'csv' file >
Expected results:
< Uploaded file is in list >
< [Upload] button becomes enabled >
< There no active validation messages >
```

Automated test preview

```
@Test(priority = 406)
    public void userCanSelectCsvFileForUploadTest() {
        databaseUploadHelper.selectUploadForDb(db1.name, db1.organization.name).selectDatabaseFile(csvFile1.path)
        def expected = [
                fileAddedToList: true,
                uploadButtonIsActive: true,
                noErrors: true
        ]
        def actual = [
                fileAddedToList: uploadDatabaseFormHelper.isFilePresentInUploadList(csvFile1.name),
                uploadButtonIsActive: !uploadDatabaseFormHelper.isUploadDisabled(),
                noErrors: !uploadDatabaseFormHelper.isErrorDisplayed()
        ]
        assertTrue(Validator.areMapsEquals(actual, expected), "Incorrect behavior after user select .csv file")
    }
```

3. Method: DataDriven (One scenario and multiple test on vary of preset data)

Example 1. (One scenario and multiple test on vary of preset data)

```
Scenario Name: < User entrance under different user roles >
< User Type = Administrator >
< User Type = Supervisor >
< User Type = Visitor >

Expected results:
< User is on ~/Administration  page >
< User is on (~/Reports) page >
< User is on (~/Main) page >
```

Example 2.(One scenario and multiple test on vary of preset data)
Scenario 1. One action and several

```
1
2  Scenario Name: < User is able to restore subscription >
3  Input data 1:
4      - < Initial subscription state: EXPIRING >
5      - < Payment method: Paypal >
6      - < New expected subsciption state: PENDING >
7  Input data 2:
8      - < Initial subscription state: CANCELLED >
9      - < Payment method: Saved >
10     - < New expected subsciption state: ACTIVE >
11
12 Expected results:
13     - On "Input data 1": < Existing subscriptions stay as they are >
       - < New добавилась новая подписка с новым статусом
```

Automation test preview...

Automated test preview

```
@Test(priority = 406, dataProvider = "provideRestartStatuses")
    public void userCanRestartSubscriptionTest(initialStatus, newStatus, paymentMethod){
        def plan = Plans.MONTH
        def user = Users.generateEndUser()
        SubscriptionUtils.prepareSubscription(initialStatus, plan, user)
        SignInUtils.signIn(user)
        paymentPageHelper.navigate().selectPlan(plan.name)
        payWithMethod(paymentMethod)
        SubscriptionFrontService subscriptionService = new SubscriptionFrontService(user)
        def subscriptionsData = subscriptionService.getSubscriptionsList().parse().data
        def expected = [
                paymentIsSuccessful: true,
                subscriptionsCount : 3,
                subscription1      : [
                        name   : Plans.FREE.fullName,
                        status: statuses.CANCELLED,
                ],
                subscription2      : [
                        name   : plan.fullName,
                        status: initialStatus,
                ],
                subscription3      : [
                        name   : plan.fullName,
                        status: newStatus,
                ]
        ]
        def actual = [
                paymentIsSuccessful: paymentPageHelper.isPaymentSuccessful(),
                subscriptionsCount : subscriptionsData.size(),
                subscription1      : [
                        name   : (subscriptionsData[0]?.planName) ?: '',
                        status: (subscriptionsData[0]?.status) ?: '',
                ],
                subscription2      : [
                        name   : (subscriptionsData[1]?.planName) ?: '',
                        status: (subscriptionsData[1]?.status) ?: '',
                ],
                subscription3      : [
                        name   : (subscriptionsData[2]?.planName) ?: '',
                        status: (subscriptionsData[2]?.status) ?: '',
```

```
                ]
        ]
        assertTrue(Validator.areMapsEquals(actual, expected), "User cannot select new plan at any time
correctly")
    }
```

Automated test preview: Input data list

```
@DataProvider
    public Object[][] provideRestartStatuses(){
        return [
                [statuses.EXPIRING, statuses.PENDING, paymentMethods.paypal],
                [statuses.CANCELLED, statuses.ACTIVE, paymentMethods.saved],
        ]
    }
```