IvanCaldwell / **ios-astronomy-uitesting**
forked from LambdaSchool/ios-astronomy-uitesting

No description, website, or topics provided. Edit

Manage topics

3 commits | 1 branch | 0 releases | 1 contributor

Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

This branch is even with LambdaSchool:master.

SpencerCurtis Add README.md with project instructions

Astronomy.xcodeproj	Add project start	
Astronomy	Add project start	
.gitignore	Add .gitignore	5 months ago
README.md	Add README.md with project instructions	5 months ago

Clone with HTTPS Use SSH
Use Git or checkout with SVN using the web URL.
https://github.com/IvanCaldwell/ios-astr
Open in Desktop | Download ZIP

README.md

UI Testing - Astronomy

Introduction

The goal of this project is to write UI tests for the given project in this respository.

Instructions

Please fork and clone this repository. It contains a modified version of the Astronomy that you should use. This project is set up so you can test the UI without having to worry about the uncertainty that network calls bring into the application. This is done by using local data when running UI tests.

Part 1 - Prepare the application for UI testing

1. Add a new UI testing target to the project.
2. In the `setUp()` method of the `AstronomyUITests` class that comes with your UI test target, create a constant for the application (`let app = XCUIApplication`)
3. Set the `app` 's `launchArguments` to `["UITesting"]` . This string is the launch argument that the application will look for to know that it use local data instead of making network calls.

Part 2 - Writing the tests

Before reading the requirements for the tests to create, remember to follow the `TestPage` outline. You should create a struct that conforms to the `TestPage` protocol for every view controller. These pages should have the following:

- **Computed properties** for every element (that you care about). This also includes adding Accessibility Identifiers to those elements where applicable. **NOTE:** accessibility identifiers have already been added to the previous sol (<) and next sol (>) bar button items that get created programmatically
- **Actions** (or interactions) that you can take using the elements you just made computed properties for.
- **Verifications** that will test for expected behavior.

Create a test for every interaction you can think of. To get you started:

- Test saving a photo
- Test viewing another sol (Note that since you're using local data, there will only be 3 sols (sols 14-16)

You should be able to write **at least** 4 tests. If you're stumped on tests to write, it can be helpful to think about how a normal user would interact with the app.

Go Further

- Create UI tests for a previous project that you've made at Lambda School that does not include networking or concurrency.

AND/OR

- For an added challenge, create UI tests for the Journal app. This will require you figure out how to use local "mock" data in order to avoid performing network calls in your tests. For some help on starting with that, refer to the following in the Astronomy project:
  - In the Resources folder, there is a folder called "UI Testing" that has the mock data.
  - The "IsUITesting.swift" file
  - From there, use "Find in Project" (Command + Shift + F) and search for "if isUITesting" to show where the project has been modified to account for using the mock data instead of performing network calls.