Cypress Framework Documentation

Project Structure

```
 cypress/
    downloads/
    e2e/
        manage_listing.spec.js
        multi_calender_dso.spec.js
        pages/
            commonPage.js
            LoginPage.js
            ManagelistingPage.js
            MulticalendarPage.js
    fixtures/
        credentials.json
    reports/
    screenshots/
    support/
        commands.js
        e2e.js
cypress.config.js
package.json
```

Overview

This Cypress framework is designed to automate end-to-end testing for a web application. It includes various test scenarios, custom commands, and a reporting mechanism to ensure comprehensive test coverage and easy result analysis.

Key Files and Classes

`cypress/e2e/pages/MulticalendarPage.js`

This file contains the `MulticalendarPage` class which includes methods to interact with the Multi Calendar page.

Example Methods:

- `multiCalendarSubMenu()`
- `oftenPopUp()`
- `listFilterDropDown()`
- `selectpms(pmsname)`
- `selecLeftList(checkboxesToClick)`
- `clickApplyOverride()`
- `dsoformTitle()`
- `requiredCalendarFields()`
- `requiredPriceSettingFields()`
- `reuquiredGreaterTenValueValidation()`
- `fixedPrice(fixedpricevalue)`
- `minPrice(minpricevalue)`
- `maxPrice(maxpricevalue)`
- `minStay(minstayvalue)`
- `addPricing()`
- `saveRefresh()`
- `progressUpdate()`
- `negativeMCScenariosOne()`
- `negativeMCScenariosTwo()`
- `functionalMCScenarios()`

`cypress/e2e/pages/ManagelistingPage.js`

This file contains the `ManagelistingPage` class which includes methods to interact with the Manage Listing page.

Example Methods:

- `managelistingSubMenu()`
- `filterHeaderText()`

- `selectByListingInfo()`
- `selectCityOptn()`
- `selectCity(citynameValue)`
- `applyFilter()`
- `verifyManageListingText()`
- `enterMinValue(value)`
- `clickMaxValue()`
- `failureMessage()`
- `locationFailureMessage()`
- `successMessage()`
- `clickEditLatLong()`
- `enterLatitude(value)`
- `updateLocation()`
- `searchBox(searchValue)`
- `noRecordFound()`
- `negativeManageListScenariosOne()`
- `negativeManageListScenariosTwo()`

`cypress/e2e/pages/commonPage.js`

This file contains the `commonPage` class which includes common methods used across different tests.

`cypress/e2e/manage_listing.spec.js`

This file contains the test cases for managing listings.

Example Test Case:

```
describe('Manage Listing Scenarios', () => {
    beforeEach(() => {
        cy.fixture('credentials').then(user => {
            cy.login(user.userDetails.username, user.userDetails.password);
        });
    });

    const performCommonSteps = () => {
        common.dynamicPricingDropDwn();
        managelisting.managelistingSubMenu();
        managelisting.filterHeaderText();
        managelisting.selectByListingInfo();
        managelisting.selectCityOptn();
        managelisting.selectCity(['chennai', 'dunes']);
        managelisting.applyFilter();
        managelisting.verifyManageListingText();
    };

    it('Manage Listing - e2e - 1', () => {
        performCommonSteps();
        managelisting.negativeManageListScenariosOne();
        managelisting.enterMinValue('312'); // **Intentional scenario: This will fail the success message assertion if you do not ch
        managelisting.clickMaxValue();
        managelisting.successMessage();
    });

    it('Manage List - e2e - 2', () => {
        performCommonSteps();
        managelisting.negativeManageListScenariosTwo();
    });
});
```

`cypress/e2e/multi_calender_dso.spec.js`

This file contains the test cases for the Multi Calendar DSO scenarios.

Example Test Case:

```
describe('DSO Scenarios', () => {
  beforeEach(() => {
    cy.fixture('credentials').then(user => {
      cy.login(user.userDetails.username, user.userDetails.password);
    });
  });

  it('Mutli calendar DSO - e2e - 1', () => {
    common.dynamicPricingDropDwn();
    multiCalendar.multiCalendarSubMenu();
    multiCalendar.oftenPopUp();
    multiCalendar.listFilterDropDown();
    multiCalendar.selectpms('VRM');
    multiCalendar.selecLeftList(2);
    multiCalendar.clickApplyOverride();
    multiCalendar.dsoformTitle();
    multiCalendar.negativeMCScenariosOne();
    multiCalendar.functionalMCScenarios();
  });

  it('Mutli calendar DSO - e2e - 2', () => {
    common.dynamicPricingDropDwn();
    multiCalendar.multiCalendarSubMenu();
    multiCalendar.oftenPopUp();
    multiCalendar.selecLeftList(2);
    multiCalendar.clickApplyOverride();
    multiCalendar.dsoformTitle();
    cy.selectDate('2025-02-7', multiCalendar);
    multiCalendar.negativeMCScenariosTwo();
    multiCalendar.saveRefresh();
    multiCalendar.progressUpdate();
  });
});
```

`cypress/fixtures/credentials.json`

This file contains the credentials used for logging in during tests.

`cypress/support/commands.js`

This file contains custom Cypress commands.

Example Custom Command:

```
Cypress.Commands.add('login', (username, password) => {
    const loginPage = new LoginPage();
    loginPage.visit('/');
    loginPage.enterUsername(username);
    loginPage.enterPassword(password);
    loginPage.submit();
    cy.url().should('include', '/pricing');
});
```

`cypress/support/e2e.js`

This file is processed and loaded automatically before your test files. It imports the custom commands.

`cypress.config.js`

This file contains the Cypress configuration.

Example Configuration:

```
const { defineConfig } = require("cypress");

module.exports = defineConfig({
  e2e: {
    setupNodeEvents(on, config) {
      // implement node event listeners here
    },
    baseUrl: "https://pricelabs.co/signin",
    specPattern: "cypress/e2e/**spec.js",
    viewportWidth: 1680,
    viewportHeight: 1050,
    pageLoadTimeout: 100000,
    requestTimeout: 100000,
    chromeWebSecurity: false,
  },
  reporter: 'cypress-mochawesome-reporter',
  reporterOptions: {
      reportDir: 'cypress/reports',
      overwrite: false,
      html: false,
      json: true
  },
});
```

`package.json`

This file contains the project metadata and dependencies.

Example:

```
{
  "name": "pricelabs",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "devDependencies": {
    "cypress": "^13.16.0",
    "cypress-mochawesome-reporter": "^3.8.2"
  }
}
```

Running Tests

To run the Cypress tests, you can use the following command:

```
npx cypress open /for headed execution
npx cypress run //for headless execution
```

This will open the Cypress Test Runner where you can select and run your test cases.

Custom Commands

`login`

Logs in a user using the provided username and password.

```
 Cypress.Commands.add('login', (username, password) => {
     const loginPage = new LoginPage();
     loginPage.visit('/');
     loginPage.enterUsername(username);
     loginPage.enterPassword(password);
     loginPage.submit();
});
```

`selectDate`

Selects a date from a date picker.

```
Cypress.Commands.add('selectDate', (userInputDate, pageInstance) => {
    const [targetYear, targetMonth, targetDay] = userInputDate.split('-');

    // Open the calendar
    cy.get(pageInstance.datePickerInputLocator).first().click();

    // Function to get month name
    const getMonthName = (month) => {
        const months = [
            'January', 'February', 'March', 'April', 'May', 'June',
            'July', 'August', 'September', 'October', 'November', 'December',
        ];
        return months[parseInt(month, 10) - 1];
    };

    // Function to select day from the calendar
    const selectDayFromCalendar = (calendarSelector, day) => {
        cy.get(`${calendarSelector} ${pageInstance.datePickerDayLocator}`)
            .contains(day)
            .click({ force: true });
        cy.get(`${calendarSelector} ${pageInstance.datePickerDayLocator}`).contains(day).click();
        cy.get(pageInstance.datePickerInputLocator)
            .first()
            .should('contain.value', day);
    };

    // Function to select month and year
    const selectMonthAndYear = () => {
        cy.get(pageInstance.datePickerHeaderLocator).then(($headers) => {
            const leftHeader = $headers.eq(0).text();
            const rightHeader = $headers.eq(1).text();

            // Check which calendar (left or right) contains the target month and year
            if (
                leftHeader.includes(`${targetYear}`) &&
                leftHeader.includes(getMonthName(targetMonth))
            ) {
                selectDayFromCalendar('.react-datepicker__month:eq(0)', targetDay);
            } else if (
                rightHeader.includes(`${targetYear}`) &&
                rightHeader.includes(getMonthName(targetMonth))
            ) {
                selectDayFromCalendar('.react-datepicker__month:eq(1)', targetDay);
            } else {
                // Navigate to next month and retry
                if (pageInstance.datePickerNextButtonLocator.length > 1) {
                    cy.get(pageInstance.datePickerNextButtonLocator).first().click();
                    selectMonthAndYear();
                }
            }
        });
    };

    selectMonthAndYear();
});
```

Test Scenarios

Functional Scenarios

1. **Manage Listing - e2e - 1**
    - Perform common steps
    - Enter minimum price value
    - Click on the maximum price input

    - Verify success message for updating min price

2. **Mutli calendar DSO** - **e2e** - **1**
    - Open dynamic pricing dropdown
    - Navigate to Multi Calendar submenu
    - Handle popup
    - Select PMS option
    - Select checkboxes in the left list
    - Apply override
    - Verify DSO form title
    - Execute functional scenarios

Negative Scenarios

1. **Manage Listing** - **e2e** - **2**
    - Perform common steps
    - Enter invalid latitude value
    - Verify failure message for invalid location data

2. **Mutli calendar DSO** - **e2e** - **2**
    - Open dynamic pricing dropdown
    - Navigate to Multi Calendar submenu
    - Handle popup
    - Select checkboxes in the left list
    - Apply override
    - Verify DSO form title
    - Select date
    - Execute negative scenarios
    - Save and refresh
    - Verify progress update

Reporting Mechanism

The framework uses `cypress-mochawesome-reporter` for generating test reports. The reports are saved in the `cypress/reports` directory.

Configuration

The reporter is configured in the `cypress.config.js` file:

```
 reporter: 'cypress-mochawesome-reporter',
reporterOptions: {
    reportDir: 'cypress/reports',
    overwrite: false,
    html: false,
    json: true
}
```

Generating Reports

To generate the reports, run the following command:

```
npx cypress run
```

The reports will be generated in the `cypress/reports` directory in JSON format. You can convert them to HTML using the `mochawesome-report-generator` tool.

Keynotes

- **Scope for Improvement**: This assignment does not cover all possible scenarios. There is a lot of scope for improvement, including adding more test cases, handling edge cases, and improving the robustness of the tests.
- **Adding Waits**: If some assertions fail, consider adding extra waits to ensure that the elements are fully loaded before performing actions or assertions.
- **Intentional Failure Scenario**: In the `Manage Listing - e2e - 1` test case, the following line is an intentional scenario to demonstrate a failure:

    ```
    managelisting.enterMinValue('312'); // **Intentional scenario: This will fail the success message assertion if you do not char
    ```

Conclusion

This document provides an overview of the Cypress framework used in this project, including the project structure, key files, custom commands, test scenarios, and the reporting mechanism.