import { test, Locator, Page } from '@playwright/test'

export default class SubstorePage {

  readonly page: Page;

  private subStoreLink: Locator;

  private wardSupply: Locator;

  private accounts: Locator;

  private pharmacy: Locator;

  private inventory: Locator;

  private inventoryRequisition: Locator;

  private consumption: Locator;

  private reports: Locator;

  private patientConsumption: Locator;

  private return: Locator;

  private stock: Locator;

  constructor(page: Page) {

    this.page = page;

    this.subStoreLink = page.locator('//span[text()="SubStore"]');

    this.wardSupply = page.locator('');

    this.accounts = page.locator('//i[text()="Accounts"]/..');

    this.pharmacy = page.locator('//a[text()=" Pharmacy "]/..');

    this.inventory = page.locator('//a[text()=" Inventory "]/..');

    this.inventoryRequisition = page.locator('//a[text()="Inventory Requisition"]/..');

    this.consumption = page.locator('');

    this.reports = page.locator('');

    this.patientConsumption = page.locator('');

    this.return = page.locator('');

    this.stock = page.locator('');

  }

  // Getter methods to access private properties

  public getPharmacy() {

    return this.page.locator('//a[text()=" Pharmacy "]/..');

  }

  public getInventory() {

    return this.page.locator('//a[text()=" Inventory "]/..');

  }

  public getAccounts() {

  }

  /\*\*

   \* @Test6 Verify all sub-modules are displayed correctly after Clicking on the "SubStore " Module.

   \*

   \* @param {Record<string, string>} data - Not used in this method but typically used to pass additional parameters if needed.

   \* @returns {Promise<void>} - Returns void; logs error if any step fails.

   \* This method verifies the visibility and interaction with the sub-modules in the Ward Supply section.

   \* It starts by waiting for a brief timeout (2 seconds) to ensure that the page elements are fully loaded.

   \* Then, it clicks on the 'Ward Supply' module to display its sub-modules.

   \* Once the sub-modules are visible, it waits for the 'Accounts' sub-module to appear on the page, ensuring it

   \*  is ready for interaction. Finally, it clicks on the 'Accounts' sub-module to verify that it can be selected

   \* and interacted with, ensuring the functionality of the Ward Supply section.

   \*/

  async verifySubModulesDisplay() {

    const counter = await this.page.locator("//a[text()='X']");

    if (counter) {

      await counter.click();

    }

    await this.subStoreLink.click();

    await this.accounts.click();

    await this.inventory.click();

    await this.pharmacy.click();

  }

  /\*\*

   \* @Test13 Verifies navigation between different tabs within the "Inventory" tab in the Substore module.

   \*

   \*

   \* Steps:

   \* 1. Navigate to the “Substore” module.

   \* 2. Click on the “Accounts” button.

   \* 3. Select the “Inventory” tab.

   \* 4. Click and navigate between different tabs within the "Inventory" tab, including:

   \*    - “Stock”

   \*    - “Inventory Requisition”

   \*    - “Consumption”

   \*    - “Reports”

   \*    - “Patient Consumption”

   \*    - “Return”

   \*

   \* Expected Result:

   \* - Ensure successful navigation to each sub-tab of the "Inventory" tab.

   \*/

  async verifyNavigationToSubStoreModule() {

    await this.subStoreLink.click();

    await this.accounts.click();

    await this.inventory.click();

  }

  /\*\*

   \* @Test13 Navigate to the Accounts section.

   \*

   \* @returns {Promise<void>} - Returns void; ensures the Accounts section is loaded.

   \*

   \* Steps:

   \* 1. Highlight the "Accounts" link or button for visual confirmation.

   \* 2. Click on the "Accounts" element to initiate navigation.

   \* 3. Wait for the page to fully load after navigation.

   \* 4. Pause execution for 1 second to ensure the section is ready for interaction.

   \*/

  async navigateToAccounts() {

    await this.accounts.click();

    await this.page.waitForTimeout(2000)

  }

  /\*\*

   \* @Test13 Verify navigation to the Stock section.

   \*

   \* @returns {Promise<void>} - Returns void; ensures the Stock page is loaded.

   \*

   \* Steps:

   \* 1. Highlight the "Stock" element for visual confirmation.

   \* 2. Click on the "Stock" link or button to initiate navigation.

   \* 3. Wait for the page to fully load after the action.

   \* 4. Pause execution for 1 second to ensure the section is ready for interaction.

   \*/

  async verifyNavigationToStock() {

  }

  /\*\*

   \* @Test13 Verify navigation to the Inventory Requisition section.

   \*

   \* @returns {Promise<void>} - Returns void; ensures the Inventory Requisition page is loaded.

   \*

   \* Steps:

   \* 1. Highlight the "Inventory Requisition" element for visual confirmation.

   \* 2. Click on the "Inventory Requisition" link or button to initiate navigation.

   \* 3. Wait for the page to fully load after the action.

   \* 4. Pause execution for 1 second to ensure the section is ready for interaction.

   \*/

  async verifyNavigationToInventoryRequisition() {

  }

  /\*\*

   \* @Test13 Verify navigation to the Consumptions section.

   \*

   \* @returns {Promise<void>} - Returns void; ensures the Consumptions page is loaded.

   \*

   \* Steps:

   \* 1. Highlight the "Consumptions" element for visual confirmation.

   \* 2. Click on the "Consumptions" link or button to initiate navigation.

   \* 3. Wait for the page to fully load after the action.

   \* 4. Pause execution for 1 second to ensure the section is ready for interaction.

   \*/

  async verifyNavigationToConsumptions() {

  }

  /\*\*

   \* @Test13 Verify navigation to the Reports section.

   \*

   \* @returns {Promise<void>} - Returns void; ensures the Reports page is loaded.

   \*

   \* Steps:

   \* 1. Highlight the "Reports" element for visual confirmation.

   \* 2. Click on the "Reports" link or button to initiate navigation.

   \* 3. Wait for the page to fully load after the action.

   \* 4. Pause execution for 1 second to ensure the section is ready for interaction.

   \*/

  async verifyNavigationToReports() {

  }

  /\*\*

   \* @Test13 Verify navigation to the Patient Consumptions section.

   \*

   \* @returns {Promise<void>} - Returns void; ensures the Patient Consumptions page is loaded.

   \*

   \* Steps:

   \* 1. Highlight the "Patient Consumptions" element for visual confirmation.

   \* 2. Click on the "Patient Consumptions" link or button to initiate navigation.

   \* 3. Wait for the page to fully load after the action.

   \* 4. Pause execution for 1 second to ensure the section is ready for interaction.

   \*/

  async verifyNavigationToPatientConsumptions() {

  }

  /\*\*

   \* @Test Verify navigation to the Return section.

   \*

   \* @returns {Promise<void>} - Returns void; ensures the Return page is loaded.

   \*

   \* Steps:

   \* 1. Highlight the "Return" element for visual confirmation.

   \* 2. Click on the "Return" link or button to initiate navigation.

   \* 3. Wait for the page to fully load after the action.

   \* 4. Pause execution for 1 second to ensure the section is ready for interaction.

   \*/

  async verifyNavigationToReturn() {

  }

  /\*\*

   \* @Test14 Captures a screenshot of the Inventory Requisition section within the Substore module.

   \* @Return The screenshot of the Inventory Requisition section.

   \*

   \*

   \* Steps:

   \* 1. Navigate to the “Substore” module.

   \* 2. Click on the “Accounts” button.

   \* 3. Select the “Inventory” tab and then the “Inventory Requisition” sub-tab.

   \* 4. Capture a screenshot of the page and save at the default location.

   \*

   \* Expected Result:

   \* - Screenshot of the page is captured and saved successfully in the specified folder.

   \*/

  async captureScreenshotOfInventoryRequisitionSection() {

    const counter = await this.page.locator("//a[text()='X']");

    if (counter) {

      await counter.click();

    }

    await this.subStoreLink.click();

    await this.accounts.click();

    await this.inventory.click()

    await this.inventoryRequisition.click();

    const folder = 'screenshots';

    const filepath = `${folder}/${'inventory-requisition-section-chromium-win32-chromium-win32.png'}`;

    const buffer = await this.page.screenshot({ path: filepath });

    return buffer;

  }

}