import { expect, test, Page } from "playwright/test";

import UtilitiesPage from "../../pages/UtilitiesPage";

import DispensaryPage from "../../pages/DispensaryPage";

import { LoginPage } from "../../pages/LoginPage";

import ProcurementPage from "../../pages/ProcurementPage";

import ADTPage from "../../pages/ADTPage";

import LaboratoryPage from "../../pages/LaboratoryPage";

import testData from "../../Data/testData.json";

import { SettingsPage } from "src/pages/SettingsPage";

import PharmacyPage from "../../pages/PharmacyPage";

import MaternityPage from "../../pages/MaternityPage";

import SubstorePage from "../../pages/SubstorePage";

import NursingPage from "../../pages/NursingPage";

import AccountingPage from "../../pages/AccountingPage";

test.describe("Yaksha", () => {

  let utilitiesPage: UtilitiesPage;

  let dispensaryPage: DispensaryPage;

  let procurementPage: ProcurementPage;

  let loginPage: LoginPage;

  let adtPage: ADTPage;

  let laboratoryPage: LaboratoryPage;

  let settingsPage: SettingsPage;

  let pharmacyPage: PharmacyPage;

  let maternityPage: MaternityPage;

  let subStorePage: SubstorePage;

  let nursingPage: NursingPage;

  let accountingPage: AccountingPage;

  test.beforeEach(async ({ page, baseURL }) => {

    await page.goto(baseURL as string);

    // Initialize page objects

    loginPage = new LoginPage(page);

    utilitiesPage = new UtilitiesPage(page);

    dispensaryPage = new DispensaryPage(page);

    procurementPage = new ProcurementPage(page);

    adtPage = new ADTPage(page);

    laboratoryPage = new LaboratoryPage(page);

    settingsPage = new SettingsPage(page);

    pharmacyPage = new PharmacyPage(page);

    maternityPage = new MaternityPage(page);

    subStorePage = new SubstorePage(page);

    nursingPage = new NursingPage(page);

    accountingPage = new AccountingPage(page);

    // Login before each test

    const validLoginData = {

      ValidUserName: testData.ValidLogin[0].ValidUserName as string,

      ValidPassword: testData.ValidLogin[1].ValidPassword as string,

    };

    await loginPage.performLogin(validLoginData);

    // Verify login was successful

    await verifyUserIsLoggedin(page);

  });

  // Individual test cases

  test("TS-1 Verify the dispensary signout button has a ToolTip", async ({

    page,

  }) => {

    const tooltipText =

      await dispensaryPage.verifyAndReturnDispensaryTooltipText();

    await verifyTooltipTextMatches(page, tooltipText);

  });

  test("TS-2 Verify Navigate to Patient Overview from Past Days Records", async ({

    page,

  }) => {

    await nursingPage.verifyPatientOverviewFromPastDaysRecords();

    await verifyVisibility(page);

  });

  test("TS-3 Verify File Upload for a Past Patient Record", async ({

    page,

  }) => {

    await nursingPage.verifyfileupload();

    await verifyfileuploadIsUploaded(page);

  });

  test.describe("runs in parallel with other describes", () => {

    test.describe.configure({ mode: "default" });

    test("TS-4 Verify Activation of BANK A/C # Ledger", async ({ page }) => {

      await accountingPage.verifyActivationLedger();

      await verifyDeactivateButtonIsVisible(page);

    });

    test("TS-5 Verify Deactivation of BANK A/C # Ledger", async ({ page }) => {

      await accountingPage.verifyDeactivationLedger();

      await verifyActivateButtonIsVisible(page);

    });

  });

  test("TS-6 Verify all sub-modules are displayed correctly after Clicking on the SubStore Module", async ({

    page,

  }) => {

    await subStorePage.verifySubModulesDisplay();

    await verifyCorrectSubModulesDisplay(subStorePage);

  });

  test("TS-7 Verify the tooltip and its text present on hover the mouse on Star", async ({

    page,

  }) => {

    const tooltipText = await maternityPage.getTooltipTextFromStar();

    await assertTooltipText(tooltipText);

  });

  test("TS-8 Verify Request for Quotation Generation", async ({ page }) => {

    await procurementPage.verifyRequestForQuotationGeneration();

    await assertQuotationGenerationMessage(procurementPage);

  });

  test("TS-9 Verify table filtering for 'Male Ward'", async ({ page }) => {

    await laboratoryPage.verifyTableFiltering();

    await verifymaleward(page);

  });

  test("TS-10 Verify to export the order section data", async ({ page }) => {

    await pharmacyPage.verifyExportOrderSectionData();

    await verifyExportOrderSectionDataSpreadsheetDownload(pharmacyPage);

  });

  test("TS-11 Verify Warning Popup for Mandatory Fields in Scheme Refund", async ({

    page,

  }) => {

    await utilitiesPage.verifyWarningPopupForMandatoryFiels();

    await assertWarningPopupMessage(utilitiesPage);

  });

  test("TS-12 Verify Price Category Enable/Disable", async ({ page }) => {

    await settingsPage.verifyDisablePriceCategory();

    await verifyEnableButtonIsVisible(page, settingsPage);

    await settingsPage.verifyEnablePriceCategory();

    await verifyEnableButtonIsVisible(page, settingsPage);

  });

  test("TS-13 Verify Navigation Between Different Tabs", async ({ page }) => {

    await subStorePage.verifyNavigationToSubStoreModule();

    await verifyCurrentURL(page, "/Home/Index#/WardSupply");

    await subStorePage.navigateToAccounts();

    await subStorePage.verifyNavigationToInventoryRequisition();

    await verifyCurrentURL(

      page,

      "/Home/Index#/WardSupply/Inventory/InventoryRequisitionList"

    );

    await subStorePage.verifyNavigationToConsumptions();

    await verifyCurrentURL(

      page,

      "/Home/Index#/WardSupply/Inventory/Consumption/ConsumptionList"

    );

    await subStorePage.verifyNavigationToReports();

    await verifyCurrentURL(page, "/Home/Index#/WardSupply/Inventory/Reports");

    await subStorePage.verifyNavigationToPatientConsumptions();

    await verifyCurrentURL(

      page,

      "/Home/Index#/WardSupply/Inventory/PatientConsumption/PatientConsumptionList"

    );

    await subStorePage.verifyNavigationToReturn();

    await verifyCurrentURL(page, "/Home/Index#/WardSupply/Inventory/Return");

    await subStorePage.verifyNavigationToStock();

    await verifyCurrentURL(page, "/Home/Index#/WardSupply/Inventory/Stock");

  });

  test("TS-14 Verify  Capture screenshot of Inventory Requisition section", async ({

    page,

  }) => {

    const screenshot =

      await subStorePage.captureScreenshotOfInventoryRequisitionSection();

    await verifyCaptureScreenshot(page, screenshot);

  });

  test("TS-15 Verify to navigate to each section which are present in the 'Inventory' sub-module", async ({

    page,

  }) => {

    await adtPage.verifyInventorySubModuleNavigation();

    await verifyDoctorErrorIsVisible(page);

  });

});

// --------------------------------------------------------------------------------------------------------------------------

async function verifyUserIsLoggedin(page: Page) {

  await page

    .locator('//li[@class="dropdown dropdown-user"]')

    .waitFor({ state: "visible", timeout: 20000 });

  expect(

    await page.locator('//li[@class="dropdown dropdown-user"]').isVisible()

  );

}

async function verifyExportOrderSectionDataSpreadsheetDownload(

  pharmacyPage: PharmacyPage

) {

  const filePath = pharmacyPage.downloadPath;

  console.log("File downloaded to:", filePath);

  expect(filePath).toBeTruthy();

}

async function assertQuotationGenerationMessage(

  procurementPage: ProcurementPage

) {

  expect(procurementPage.quotationMessageText).toBe(

    "Request For Quotation is Generated and Saved"

  );

}

async function assertTooltipText(actualText: string | null) {

  expect(actualText).not.toBeNull();

  expect(actualText?.trim()).toBe("Remember this Date");

}

async function verifyCorrectSubModulesDisplay(subStorePage: SubstorePage) {

  await subStorePage.getPharmacy().waitFor({ state: "visible" });

  await subStorePage.getPharmacy().click();

  await expect(subStorePage.getPharmacy()).toBeVisible();

  await subStorePage.getInventory().click();

  await expect(subStorePage.getInventory()).toBeVisible();

}

async function verifyAlert(page: Page) {

  const alerts = page.locator('[class="main-message"]');

  const alertCount = await alerts.count();

  for (let i = 0; i < alertCount; i++) {

    const currentAlert = alerts.nth(i);

    await expect(currentAlert).toBeVisible();

  }

}

async function verifyEnableButtonIsVisible(

  page: Page,

  settingsPage: SettingsPage

) {

  await page.waitForTimeout(3000);

  await expect(settingsPage.activate).toBeVisible();

}

async function verifyDisableButtonIsVisible(

  page: Page,

  settingsPage: SettingsPage

) {

  await page.waitForTimeout(3000);

  await expect(settingsPage.disable).toBeVisible();

}

async function verifyDeactivateButtonIsVisible(page: Page) {

  await page.waitForTimeout(3000);

  const deactivateButton = await page.locator("//a[text() ='Deactivate']");

  await expect(deactivateButton).toBeVisible();

}

async function verifyActivateButtonIsVisible(page: Page) {

  await page.waitForTimeout(3000);

  const activateButton = await page.locator('//a[text()="Activate"]');

  await expect(activateButton).toBeVisible();

}

async function verifymaleward(page: Page) {

  const alert = page.locator(

    '(//div[@col-id="WardName" and contains(text(),"Male Ward")])[1]'

  );

  expect(await alert.isVisible()).toBeTruthy();

}

async function verifyVisibility(page: Page) {

  const labs = page.locator('(//div[@class="caption custom-caption"])[1]');

  expect(await labs.isVisible()).toBeTruthy();

}

async function verifyDoctorErrorIsVisible(page: Page) {

  const doctorError = page.locator("//span[@class='color-red']");

  expect(await doctorError.isVisible()).toBeTruthy();

}

async function assertWarningPopupMessage(utilitiesPage: UtilitiesPage) {

  await expect(utilitiesPage.utilities.warningPopup).toBeVisible();

}

async function verifyCaptureScreenshot(page: Page, screenshot?: Buffer) {

  const fileName = "inventory-requisition-section-chromium-win32.png";

  await expect(screenshot).toMatchSnapshot(

    "../../inventory-requisition-section-chromium-win32.png",

    {

      threshold: 1,

    }

  );

}

async function verifyfileuploadIsUploaded(page: Page) {

  await expect(

    page.locator('//p[@class="main-message" and text()=" File Uploded"]')

  ).toBeVisible();

}

async function verifyCurrentURL(page: Page, urlToMatchWith: string) {

  const currentURL = page.url();

  expect(currentURL).toContain(urlToMatchWith);

  console.log("Current URL:", currentURL);

}

async function verifyTooltipTextMatches(page: Page, tooltipText: string) {

  const expectedText =

    "You are currently in Main Dispensary dispensary. To change, you can always click here.";

  expect(tooltipText).toBe(expectedText);

}