Test Scripts for Scopex Money Mobile App

(JavaScript/TypeScript with Appium)

Test Scripts for Scopex Money Mobile App 2
(JavaScript/TypeScript with Appium) 2

Test Scripts for Scopex Money Mobile App

(JavaScript/TypeScript with Appium)

Prerequisites

Ensure you have the following set up:
- Node.js installed
- Appium server running
- WebDriverIO or any preferred framework for mobile automation

1. User Registration

Positive Scenarios

Test Script: Successful Registration with Valid Inputs

```typescript
describe('User Registration', () => {
  it('should register successfully with valid inputs', async () => {
    await driver.launchApp('Scopex Money');

    await driver.$('~Registration').click(); // Navigate to registration
    await driver.$('~name_field').setValue('John Doe');
    await driver.$('~email_field').setValue('john.doe@example.com');
    await driver.$('~password_field').setValue('StrongPass123!');
    await driver.$('~confirm_password_field').setValue('StrongPass123!');
    await driver.$('~terms_checkbox').click(); // Agree to terms

    await driver.$('~register_button').click();

    const isDashboard = await driver.getUrl() === 'dashboard';
    expect(isDashboard).toBe(true); // Assert redirection to dashboard
  });
});
```

Negative Scenarios

Test Script: Invalid Email Format

```typescript
it('should show error for invalid email format', async () => {
  await driver.launchApp('Scopex Money');
  await driver.$('~Registration').click();

  await driver.$('~name_field').setValue('Jane Doe');
```

```
  await driver.$('~email_field').setValue('jane@domain'); // Invalid email
  await driver.$('~password_field').setValue('StrongPass123!');
  await driver.$('~confirm_password_field').setValue('StrongPass123!');

  await driver.$('~register_button').click();

  const errorMessage = await driver.$('~error_message').getText();
  expect(errorMessage).toBe('Invalid email format.'); // Assert error message
});
```

Test Script: Email Already Registered

```typescript
it('should show error for already registered email', async () => {
  await driver.launchApp('Scopex Money');
  await driver.$('~Registration').click();

  await driver.$('~name_field').setValue('Alice Smith');
  await driver.$('~email_field').setValue('john.doe@example.com'); // Existing email
  await driver.$('~password_field').setValue('StrongPass123!');
  await driver.$('~confirm_password_field').setValue('StrongPass123!');

  await driver.$('~register_button').click();

  const errorMessage = await driver.$('~error_message').getText();
  expect(errorMessage).toBe('Email already registered.'); // Assert error message
});
```

Test Script: Weak Password

```typescript
it('should show error for weak password', async () => {
  await driver.launchApp('Scopex Money');
  await driver.$('~Registration').click();

  await driver.$('~name_field').setValue('Bob Brown');
  await driver.$('~email_field').setValue('bob.brown@example.com');
  await driver.$('~password_field').setValue('weak'); // Weak password
  await driver.$('~confirm_password_field').setValue('weak');

  await driver.$('~register_button').click();

  const errorMessage = await driver.$('~error_message').getText();
  expect(errorMessage).toBe("Password does not meet security criteria."); // Assert error message
});
```

Test Script: Missing Required Fields

```typescript
it("should show error for missing required fields", async () => {
  await driver.launchApp("Scopex Money");
  await driver.$("~Registration").click();
```

```
  await driver.$("~name_field").setValue(""); // Missing name
  await driver.$("~email_field").setValue("");

  await driver.$("~register_button").click();

  const errorMessage = await driver.$("~error_message").getText();
  expect(errorMessage).toBe("Name and email are required fields."); // Assert error message
});
```

Test Script: Non-Matching Passwords

```typescript
it("should show error for non-matching passwords", async () => {
  await driver.launchApp("Scopex Money");
  await driver.$("~Registration").click();

  await driver.$("~name_field").setValue("Charlie Green");
  await driver.$("~email_field").setValue("charlie.green@example.com");
  await driver.$("~password_field").setValue("StrongPass123!");
  await driver.$("~confirm_password_field").setValue("DifferentPass456!");

  await driver.$("~register_button").click();

  const errorMessage = await driver.$("~error_message").getText();
  expect(errorMessage).toBe("Passwords do not match."); // Assert error message
});
```

Test Script: Terms and Conditions Not Accepted

```typescript
it("should show error when terms are not accepted", async () => {
  await driver.launchApp("Scopex Money");
  await driver.$("~Registration").click();

  await driver.$("~name_field").setValue("Diana White");
  await driver.$("~email_field").setValue("diana.white@example.com");
  await driver.$("~password_field").setValue("StrongPass123!");

  // Do not check terms checkbox

  await driver.$("~register_button").click();

  const errorMessage = await driver.$("~error_message").getText();
  expect(errorMessage).toBe("You must accept the terms and conditions."); // Assert error message
});
```

Test Script: KYC Document Issues

```typescript
it("should show error for invalid KYC document", async () => {
  await driver.launchApp("Scopex Money");
```

```
  await driver.$("~Registration").click();

  await driver.$("~name_field").setValue("Ethan Black");
  await driver.$("~email_field").setValue("ethan.black@example.com");
  await driver.$("~password_field").setValue("StrongPass123!");

  // Simulate uploading invalid KYC document (this may vary based on your implementation)
  // Assuming there's a method to upload files in your app's context.
  uploadFileToKYCDocumentField(driver, 'invalid_document.pdf');

  await driver.$("~register_button").click();

  const errorMessage = await driver.$("~error_message").getText();
  expect(errorMessage).toBe("Invalid KYC document."); // Assert error message
});
```

2. Adding a Recipient

Positive Scenarios

Test Script: Successful Addition of a Recipient

```typescript
describe('Adding a Recipient', () => {
  it('should add recipient successfully', async () => {
    await driver.launchApp('Scopex Money');
    loginAsValidUser(); // Assuming a function that logs in a valid user

    // Navigate to Add Recipient section
    await driver.$(~'Add Recipient').click();
    await driver.$(~'recipient_name_field').setValue(~'John Doe');
    await driver.$(~'recipient_account_number_field').setValue(~'1234567890');

    // Click Add button
    await driver.clickButton(~'add_recipient_button');

    const successMessage = awai.driver.getText(~'success_message');
    expect(successMessage).toBe(~'Recipient added successfully!');

    // Validate recipient appears in recipient list after addition
    navigateToRecipientList();
    const isRecipientInList = awai.driver.isElementDisplayed(~'John Doe');
    expect(isRecipientInList).toBe(true);
  });
});
```

Negative Scenarios

Test Script: Duplicate Recipient

```typescript
it(`should show error for duplicate recipient`, async () => {
    launchApp(`Scopex Money`);
    loginAsValidUser();

    navigateTo(`Add Recipient`);

    enterText(`recipient_name_field`, `John Doe`);
    enterText(`recipient_account_number_field`, `1234567890`);

    clickButton(`add_recipient_button`);

    const errorMessage = getText(`error_message`);
    expect(errorMessage).toBe(`Recipient already exists.`);
});
```

Test Script: Invalid Recipient Details

```typescript
it(`should show error for invalid recipient details`, async () => {
    launchApp(`Scopex Money`);
    loginAsValidUser();

    navigateTo(`Add Recipient`);

    enterText(`recipient_name_field`, ``);
    enterText(`recipient_account_number_field`, `invalid_account`);

    clickButton(`add_recipient_button`);

    const errorMessage = getText(`error_message`);
    expect(errorMessage).toBe(`Name and account number are required fields.`);
});
```

Test Script: Exceeding Character Limits

```typescript
it(`should show error for exceeding character limits`, async () => {
    launchApp(`Scopex Money`);

    navigateTo(`Add Recipient`);

    const longName = `A`.repeat(256);
    enterText(`recipient_name_field`, longName);

    clickButton(`add_recipient_button`);

    const errorMessage = getText(`error_message`);
    expect(errorMessage).toBe(`Name exceeds character limit.`);
});
```

Test Script: Network Issues

```typescript
it(`should show error during network issues`, async () => {
    simulateNetworkFailure();

    launchApp(`Scopex Money`);

    loginAsValidUser();

    navigateTo(`Add Recipient`);

    enterText(`recipient_name_field`, `John Doe`);
    enterText(`recipient_account_number_field`, `1234567890`);

    clickButton(`add_recipient_button`);

    const errorMessage = getText(`error_message`);
    expect(errorMessage).toBe(`Network error. Please try again later.`);
});
```

Test Script: Unauthorized Access

```typescript
it(`should redirect to login screen when trying to add recipient without login`, async () => {
    launchApp(`Scopex Money`);

    logout();

    navigateTo(`Add Recipient`);

    const isRedirectedToLogin = getUrl() === `login_screen`;
    expect(isRedirectedToLogin).toBe(true);
});
```

3. Logout

Positive Scenarios

Test Script: Successful Logout Redirects to Login Screen

```typescript
describe(`Logout`, () => {
  it(`should log out successfully and redirect to login screen`, async () => {
     launchApp(`Scopex Money`);

     loginAsValidUser();
```

```
    clickButton(`logout_button`);

    const isRedirectedToLogin = getUrl() === `login_screen`;
    expect(isRedirectedToLogin).toBe(true);
  });
});
```

Negative Scenarios

Test Script: Session Timeout

```typescript
it(`should handle session timeout during logout`, async () => {
    launchApp(`Scopex Money`);

    loginAsValidUser();

    simulateSessionTimeout();

    clickButton(`logout_button`);

    const errorMessage = getText(`error_message`);
    expect(errorMessage).toBe(`Session has expired. Please log in again.`);
});
```

Test Script: Concurrent Session Logout

```typescript
it(`should manage concurrent session logout correctly`, async () => {
    launchApp(`Scopex Money`);

    loginAsValidUser();

    openNewDeviceAndLogin();

    clickButton(`logout_button`);

    const isRedirectedToLogin = getUrl() === `login_screen`;
    expect(isRedirectedToLogin).toBe(true);

    switchBackToFirstDevice();

    const isFirstDeviceRedirected = getUrl() === `login_screen`;
    expect(isFirstDeviceRedirected).toBe(true);
});
```

Test Script: Access Restricted Pages Post Logout

```typescript
it(`should redirect to login screen when accessing restricted pages post logout`, async () => {
```

```
    launchApp(`Scopex Money`);

    loginAsValidUser();

    clickButton(`logout_button`);

    navigateTo(`home_screen`);

    const isRedirectedToLogin = getUrl() === `login_screen`;
    expect(isRedirectedToLogin).toBe(true);
});
```

Test Script: Logout Without Being Logged In

```typescript
it(`should handle logout attempt when not logged in`, async () => {
    launchApp(`Scopex Money`);

    clickButton(`logout_button`);

    const errorMessage = getText(`error_message`);
    expect(errorMessage).toBe(`You are not logged in.`);
});
```