QA Task Assignment: Manual & Automation Testing for Scopex Money

Part 1:

1. Test Approach:

The test approach will focus on validating the core functionalities of **User Registration**, **Adding a Recipient**, and **Logout**. The primary goal is to ensure these features work as intended in the application.

---

2. Key Test Scenarios:

A. User Registration:

- **Test Scenario 1**: Valid User Registration (all required fields are correctly filled).
- **Test Scenario 2**: Invalid Email Address (invalid email format).
- **Test Scenario 3**: Password Requirements (password too short or missing special characters).
- **Test Scenario 4**: Missing Required Fields (empty fields like first name, email, etc.).
- **Test Scenario 5**: Duplicate Email Registration (trying to register with an already existing email).
- **Test Scenario 6**: User Registration with valid and invalid phone number formats.
- **Test Scenario 7**: Successful registration with acceptance of terms and conditions.

B. Adding a Recipient:

- **Test Scenario 1**: Add a Recipient with all required fields (name, bank account, address, etc.).
- **Test Scenario 2**: Add a Recipient with missing required fields (e.g., no bank account number).
- **Test Scenario 3**: Add a Recipient with invalid bank account details.
- **Test Scenario 4**: Add a Recipient with valid but different country formats.

- **Test Scenario 5**: Add a duplicate recipient.
- **Test Scenario 6**: Validating the system response to adding a recipient with special characters.

C. Logout:

- **Test Scenario 1**: Successfully log out from the system.
- **Test Scenario 2**: Verify if the session is terminated after logout (The user is unable to access pages without logging in again).
- **Test Scenario 3**: Check if the system maintains the state of a session if not logged out properly (checking security).
- **Test Scenario 4**: Logout on multiple devices or browsers and ensure successful logout.

---

3. Test Data Requirements and Preconditions:

A. Test Data for User Registration:

- Valid email addresses, passwords, and full name.
- Invalid email (e.g., missing “@” or domain name).
- Duplicate email for testing registration failure.
- Invalid password (e.g., less than 8 characters, no special characters).
- Test accounts with valid and invalid phone numbers.

B. Test Data for Adding a Recipient:

- Valid recipient details, including name, address, and bank account number.
- Invalid recipient details (e.g., invalid account number or address format).
- Duplicate recipient details for error validation.
- Special characters in the recipient's name or address.

C. Test Data for Logout:

- Ensure that the user is logged into the system beforehand.
- Test with multiple simultaneous log in sessions (i.e., across multiple browsers or devices).

---

4. Expected Outcomes:

A. User Registration:

- **Valid registration:** A success message with redirection to the dashboard.
- **Invalid registration (e.g., invalid email/password):** Error message displayed next to the corresponding field.
- **Duplicate email registration:** Display an error message like "Email already in use".
- **Empty fields:** An error message indicating which fields are missing.
- **Valid phone number:** Proper format validation, and appropriate error messages for invalid phone numbers.

B. Adding a Recipient:

- **Valid recipient data:** Recipient added successfully with a confirmation message.
- **Missing fields:** Error message indicating which field is missing or invalid.
- **Invalid bank account data:** The system should reject and display an error.
- **Duplicate recipient:** An alert that the recipient already exists.

C. Logout:

- **Successful logout:** The user should be redirected to the login page, and the session should be terminated.
- **Post-logout:** Attempting to access user pages without logging in should redirect to the login page.

---

5. Test Environment:

The tests will be conducted on the following devices, browsers, and OS configurations to ensure wide compatibility:

A. Devices:

- Desktop (Windows, Mac)
- Mobile (Android, iOS)

B. Browsers:

- Google Chrome

C. OS Versions:

- **Windows 10** (Chrome, Firefox, Edge)

---

6. Assumptions:

- The system is integrated with a working backend for user data management and recipient information.
- The user is aware of the website/application before testing and knows basic navigation.
- All UI components are fully developed and are not part of the testing scope unless directly related to functionality.

---

7. Risks and Constraints:

A. Risks:

- **Integration Issues:** If the application is not fully integrated with the backend, tests may fail or not return the expected results.
- **Device/Browser Compatibility:** Some features may not behave identically across all devices or browsers.
- **Environmental Issues:** Changes in the production environment, such as security updates or patches, may cause test failures.

B. Constraints:

- Limited access to test environments may delay test execution.
- Limited availability of actual data for realistic test cases (e.g., duplicate email addresses).

1. Set Up Appium Configuration (config/appium.config.ts):

```
import { Capabilities } from 'appium';

export const appiumConfig = {
  platformName: 'Android', // Change to 'iOS' for iOS testing
  platformVersion: '11.0',
  deviceName: 'Android Emulator', // Device or real device name
  app: 'path/to/your/app.apk', // Path to the app
  automationName: 'UiAutomator2',
  noReset: true,
  fullContextList: true,
  clearSystemFiles: true,
};
```

2. Define Page Object Models (POM):

- **Login Page (pages/login.page.ts):**

```
import { driver } from 'wd'; // Assuming you're using WebDriverIO/WD
import { By } from 'selenium-webdriver';

export class LoginPage {
  usernameField = By.id('com.scopex.money:id/username');
  passwordField = By.id('com.scopex.money:id/password');
  loginButton = By.id('com.scopex.money:id/login');

  async login(username: string, password: string) {
    await driver.elementSendKeys(this.usernameField, username);
    await driver.elementSendKeys(this.passwordField, password);
    await driver.elementClick(this.loginButton);
  }
}
```

- **Recipient Page (pages/recipient.page.ts):**

```
import { driver } from 'wd';
import { By } from 'selenium-webdriver';
```

```
export class RecipientPage {
  addRecipientButton = By.id('com.scopex.money:id/addRecipient');
  recipientNameField = By.id('com.scopex.money:id/recipientName');
  recipientPhoneField = By.id('com.scopex.money:id/recipientPhone');
  saveButton = By.id('com.scopex.money:id/save');

  async addRecipient(name: string, phone: string) {
    await driver.elementClick(this.addRecipientButton);
    await driver.elementSendKeys(this.recipientNameField, name);
    await driver.elementSendKeys(this.recipientPhoneField, phone);
    await driver.elementClick(this.saveButton);
  }
}
```

- **Home (Dashboard) Page (pages/home.page.ts):**

```
import { driver } from 'wd';
import { By } from 'selenium-webdriver';

export class HomePage {
  logoutButton = By.id('com.scopex.money:id/logout');

  async logout() {
    await driver.elementClick(this.logoutButton);
  }
}
```

- **Login Test (tests/login.spec.ts):**

```
import { expect } from 'chai';
import { LoginPage } from '../pages/login.page';
import { HomePage } from '../pages/home.page';
import { appiumConfig } from '../config/appium.config';
import { driver } from 'wd';

describe('Login Test', () => {
```

```
  let loginPage: LoginPage;
  let homePage: HomePage;

  before(async () => {
    await driver.init(appiumConfig);
    loginPage = new LoginPage();
    homePage = new HomePage();
  });

  it('should log in successfully with valid credentials', async () => {
    await loginPage.login('testuser', 'validpassword');
    const isDashboardVisible = await
driver.isElementPresent(homePage.logoutButton);
    expect(isDashboardVisible).to.be.true;
  });

  after(async () => {
    await driver.quit();
  });
});
```

- **Add Recipient Test (tests/addRecipient.spec.ts):**

```
import { expect } from 'chai';
import { RecipientPage } from '../pages/recipient.page';

describe('Add Recipient Test', () => {
  let recipientPage: RecipientPage;

  before(async () => {
    recipientPage = new RecipientPage();
  });

  it('should add a recipient successfully', async () => {
    await recipientPage.addRecipient('John Doe', '1234567890');
```

```
    const isRecipientAdded = await
driver.isElementPresent(By.xpath('//android.widget.TextView[@text="Joh
n Doe"]'));
    expect(isRecipientAdded).to.be.true;
  });
});
```

- **Logout Test (tests/logout.spec.ts):**

```
import { expect } from 'chai';
import { HomePage } from '../pages/home.page';

describe('Logout Test', () => {
  let homePage: HomePage;

  before(async () => {
    homePage = new HomePage();
  });

  it('should log out successfully', async () => {
    await homePage.logout();
    const isLoginPageVisible = await
driver.isElementPresent(By.id('com.scopex.money:id/login'));
    expect(isLoginPageVisible).to.be.true;
  });
});
```

- **Logging (utils/logger.ts):**

```
import winston from 'winston';

const logger = winston.createLogger({
  level: 'info',
  format: winston.format.simple(),
  transports: [new winston.transports.Console()],
});
```

```
export default logger;
```

- **Screenshots (utils/screenshot.ts):**

```
import { driver } from 'wd';
import fs from 'fs';

export const captureScreenshot = async (testName: string) => {
  const screenshot = await driver.takeScreenshot();
  fs.writeFileSync(`./screenshots/${testName}.png`, screenshot,
'base64');
};
```

Part 3: Bug Finding & Reporting

Bug ID: BUG-001

Title: User Registration Fails with Invalid Email Format

Environment:

- **Device:** iPhone 13
- **OS:** iOS 15.3
- **App Version:** Scopex Money App v2.0.1

Steps to Reproduce:

1. Open the Scopex Money mobile app.
2. Navigate to the Registration screen.
3. Enter an invalid email address (e.g., `user@domain` without a valid domain).
4. Enter a valid password.
5. Press the **Register** button.

Expected Result:

- The app should display an error message indicating an invalid email format, and the registration should not proceed.

Actual Result:

- The app proceeds with registration, and the user is redirected to the dashboard even with the invalid email format.

Screenshots/Videos:

Screenshot showing successful registration with an invalid email.

Severity: High

Additional Notes:

- The app should validate the email format before allowing registration. This issue affects user registration and should prevent users from creating accounts with invalid email addresses.

Bug ID: BUG-002

Title: Adding a Recipient with Missing Fields Causes App Freeze

Environment:

- **Device:** Samsung Galaxy S21
- **OS:** Android 12
- **App Version:** Scopex Money App v2.1.0

Steps to Reproduce:

1. Open the Scopex Money app on the device.
2. Navigate to the **Recipient** section.
3. Click on the **Add Recipient** button.
4. Leave the **Phone Number** field empty and fill in the **Name** field.
5. Tap on the **Save** button.

Expected Result:

- The app should show an error message indicating that the **Phone Number** field is required.

Actual Result:

- The app freezes and becomes unresponsive. The user cannot proceed to add the recipient.

Screenshots/Videos:

Video of app freezing after trying to save with incomplete data.

Severity: Critical

Additional Notes:

- This issue prevents the user from adding a recipient if required fields are left blank. It is a major blocker for the recipient functionality and must be fixed.

Bug ID: BUG-003

Title: Logout Does Not Redirect to Login Screen on Website

Environment:

- **Device:** Laptop
- **OS:** Windows 10
- **Browser:** Google Chrome 89
- **App Version:** Scopex Money Web v2.0.0

Steps to Reproduce:

1. Open the Scopex Money website.
2. Log in with a valid user account.
3. Click on the **Logout** button on the top-right corner.
4. Wait for redirection.

Expected Result:

- The user should be redirected to the login screen after clicking **Logout**.

Actual Result:

- The user remains on the dashboard page after clicking **Logout** and is still able to access the recipient list and other restricted pages.

Screenshots/Videos:

Screenshot of the dashboard still visible after logging out.

Severity: Medium

Additional Notes:

- This issue only occurs on the website version. The app version behaves as expected. This bug impacts security and the user's ability to fully log out from the web platform.

Bug ID: BUG-004

Title: "Add Recipient" Button Unresponsive on Mobile App

Environment:

- **Device:** iPhone 11
- **OS:** iOS 14.5
- **App Version:** Scopex Money App v2.1.0

Steps to Reproduce:

1. Launch the Scopex Money app.
2. Navigate to the **Recipient** section.
3. Tap the **Add Recipient** button.

Expected Result:

- The **Add Recipient** screen should open, allowing the user to input recipient details.

Actual Result:

- The **Add Recipient** button is unresponsive. Nothing happens when the user taps the button.

Screenshots/Videos:

Screenshot of the unresponsive Add Recipient button.

Severity: High

Additional Notes:

- This issue prevents users from adding new recipients. It should be investigated, as it blocks a key feature of the app.