**Appium Testing Milestone Assessment — Real-Time Scenario-Based Case Study**

**Scenario: E-Commerce Shopping App Testing**

**Objective:** Automate testing for a seamless user journey in an e-commerce mobile application using Appium.

**Key Screens:**

1. **Screen 1: Login/Sign-Up Screen**
   * **Details:** Users can log in using their existing credentials or sign up with a new account by providing an email, password, and mobile number.
   * **Scenario:**
     + Validate correct input fields for credentials and login functionality.
     + Check error handling for invalid credentials.
     + Test UI elements like "Forgot Password?" and "Sign Up" links.
2. **Screen 2: Product Listing Page**
   * **Details:** Display of various products with filters, sort options, and product categories.
   * **Scenario:**
     + Validate scrolling through the product list without glitches.
     + Check if sorting and filtering yield accurate results.
     + Test for edge cases where the product list might be empty.
3. **Screen 3: Checkout Screen**
   * **Details:** A final checkout screen showing selected products, their quantities, and total prices, with options for payment and delivery.
   * **Scenario:**
     + Validate the correctness of the total price calculation.
     + Test item removal functionality and quantity updates.
     + Check payment method selection and form validations.

**App Scenario: Health & Fitness Tracker App (3 Screens)**

**1. Splash Screen:**

* Upon launching the app, the Splash screen displays the app logo for 3 seconds.
* After the delay, it navigates to the **User Onboarding Screen**.
* Verify if the app handles the splash duration and navigation correctly.

**Test Case Scenario:**

* Ensure the splash screen is displayed for exactly 3 seconds.
* Verify that the splash screen transitions to the onboarding screen without glitches.
* Ensure no crashes occur during this transition.

**2. User Onboarding Screen:**

* The onboarding screen displays 3 informational slides with swipe functionality.
* Each slide contains fitness tips.
* There’s a "Skip" button to navigate directly to the **Dashboard Screen**.

**Test Case Scenario:**

* Validate swipe gestures and ensure smooth transitions between slides.
* Test that the "Skip" button navigates correctly to the dashboard screen.
* Confirm that the completion of all slides also navigates to the dashboard screen.

**3. Dashboard Screen:**

* Displays user fitness metrics (steps taken, calories burned, etc.)
* Contains a "Sync Device" button to sync data from a wearable device.
* A success message "Data Synced Successfully" should appear upon completion.

**Test Case Scenario:**

* Verify UI elements for proper visibility and accurate data display.
* Test the "Sync Device" button functionality and ensure successful syncing.
* Confirm the success message appears after a successful sync.

**App Scenario: Food Delivery App (3 Screens)**

**Scenario:**  
Create and test a real-time **Food Delivery App** to ensure its critical features work seamlessly on Android and iOS devices. The app allows users to browse restaurants, view menu items, and place orders.

**Screen 1: Home Screen**

* Display a list of nearby restaurants fetched from a backend API.
* Each restaurant should show its name, image, rating, and delivery time.
* Users can scroll the list and select any restaurant to navigate to its menu.

**Test Objectives:**

1. Verify that the list of restaurants is displayed correctly and matches the API response.
2. Check the scrolling functionality and ensure smooth navigation.
3. Test that tapping a restaurant navigates the user to the Menu screen.

**Screen 2: Menu Screen**

* Show a list of food items with their names, images, prices, and "Add to Cart" buttons.
* A floating cart icon should display the number of items added.
* Clicking the cart icon navigates to the Order Summary screen.

**Test Objectives:**

1. Validate that all menu items load correctly with accurate details.
2. Test the "Add to Cart" functionality and verify the cart counter updates properly.
3. Check if tapping the floating cart button navigates the user to the Order Summary screen.

**Screen 3: Order Summary Screen**

* Display the list of selected food items with names, quantities, and prices.
* Show the total order amount.
* Include a "Place Order" button to confirm the order.

**Test Objectives:**

1. Verify that all selected items appear correctly with the right quantities and prices.
2. Ensure the total amount is calculated accurately.
3. Test the functionality of the "Place Order" button.

.

### **Case Study: Cake Recipe Finder App(https://www.themealdb.com/api/json/v1/1/search.php?s=cake)Objective**

Develop an Android app using **Java/Kotlin and XML** that fetches cake recipes from TheMealDB API and displays them in a structured format with a **smooth user experience**. The app will have **four screens**:

1. **Recipe List Screen** – Fetches and displays cake recipes in a RecyclerView.
2. **Recipe Detail Screen** – Shows detailed recipe information when a user selects an item.
3. **Favorites Screen** – Allows users to save and view favorite recipes using Room Database.

## **📱 App Flow**

### **1 Recipe List Screen**

* Uses **Retrofit** to fetch data from **TheMealDB API**.
* Shows the recipes in a **RecyclerView** with images and names.
* Clicking on a recipe navigates to the **Recipe Detail Screen**.

### **2 Recipe Detail Screen**

* Displays detailed information:
  + Recipe **image, name, ingredients, instructions**.
  + A **"Save to Favorites"** button to store the recipe locally.
* Uses **Room Database** to save and retrieve favorite recipes.

### **3 Favorites Screen**

* Displays saved recipes from **Room Database**.
* Users can **remove** recipes from favorites.
* Uses a **RecyclerView** to show saved recipes.

## **🛠️ Tech Stack**

* **Android (Java & XML)**
* **Retrofit** for API calls
* **Glide/Picasso** for image loading
* **RecyclerView** for list display
* **Room Database** for favorites
* **ViewModel & LiveData** for state management

## **📝 Appium Testing Instructions**

### **1️⃣ Prerequisites**

* Install **Appium Server**
* Install **Appium Inspector**
* Set up **Android Emulator** or connect a real device
* Ensure the **Appium Java Client** library is included

### **2️⃣ Test Cases**

#### **🔹 Test 1: Fetch & Display Recipe List**

* Wait for RecyclerView using findElementById("com.example.app:id/recipe\_recyclerview")
* Verify if at least **one item** is loaded in the RecyclerView

#### **🔹 Test 2: Click on a Recipe & Open Detail Screen**

* Use findElementByXPath("//android.widget.TextView[@text='Chocolate Cake']") to click a specific recipe
* Verify the **detail screen** appears using findElementById("com.example.app:id/recipe\_image")

#### **🔹 Test 3: Add Recipe to Favorites & Verify**

* Click **"Save to Favorites"** button using findElementById("com.example.app:id/save\_button")
* Navigate to **Favorites Screen**
* Verify the saved recipe appears in RecyclerView

#### **🔹 Test 4: Remove Recipe from Favorites**

* Locate the saved recipe
* Click the **Remove Button**
* Verify if the list updates