**Project name:** Skybot Mobile Application Version 2.0

**Type of testing:** Non-Functional

**Network:** Wi-fi, 5G,4G mobile Kyivstar

**Environment:** Android 14,

Version Software A536BXXSBEXH1,

Version One UI 6.1

Screen size “6,5”

Android 12,

Model: Teclast T50\_EEA

V1.06\_20230824

Screen size “11”

**0. Installing and Uninstalling an App**

| **ID** | **Title** | **Description** | **Precondition** | **Test steps** | **Expected result** | **Status** | **Notes** | **Priority** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0.1** | **Download the Mobile App** | Verify the mobile app download successfully | The user is logged into the Google Play Store.  Sufficient free storage space available on the device | ***Method 1:***  1. Open the Google Play Store on the device.  2. In the search bar, enter the name of the app Spaces by Wix.  3. Tap on the appropriate app from the search results.  4. Tap the "Install" button.  5. Wait for the app to download and install.  6. Once installation is complete, tap the "Open" button.  ***Method 2.***  1. Follow by the instruction of document <https://docs.google.com/document/d/1mYTKMs_j1lH0dGf2_av0hNdvhBni_HdAJHzyli3RJ_Q/edit?tab=t.0#heading=h.jtz6b0s59c9t> | The app is successfully downloaded on the device | Pass |  | High |
| **0.2** | **Launching the Mobile App** | Verify the mobile app launches successfully | The app is installed on the device | 1. Open the mobile app  2. Observe the page | The app opens without errors, displaying the main screen | Pass |  | High |
| **0.3** | **Uninstalling an App from the Mobile Device** | Verify the user is able to successfully uninstall the application from their mobile device | The app is installed on the mobile device.  The user has access to the device's settings or app drawer | ***Method 1:*** Uninstall via App Drawer  1. Navigate to the App Drawer (list of all installed apps).  2. Locate the app to be uninstalled.  3. Press and hold the app icon.  4. From the options that appear, select Uninstall.  5. Confirm the action by tapping OK or Uninstall on the confirmation pop-up  ***Method 2:*** Uninstall via Device Settings  1. Open the Settings app on the device.  2. Navigate to Apps or Applications.  3. Scroll through the list and select the app to uninstall.  4. Tap Uninstall.  5. Confirm the action by tapping OK or Uninstall on the confirmation pop-up | The app is successfully uninstalled from the device.  The app is no longer available in the app drawer or in the device’s list of installed applications.  The user may receive a confirmation message that the app was uninstalled. | Pass |  | High |

**12. Performance Testing**

| **ID** | **Title** | **Description** | **Precondition** | **Test steps** | **Expected result** | **Status** | **Notes** | **Priority** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **12.1** | **Application Load Time Verification** | Assess the time required to fully load the application from the home screen | The application is installed on the device | 1. On the device's home screen, tap the app icon.  2. Measure the time from tapping the icon until the main screen of the app is fully loaded | The application loads within 2 seconds after tapping the icon | Pass |  | High |
| **12.2** | **Application Response Time Verification** | Measure the application’s response time to key user actions for smooth performance | The application is open and in a ready state | 1. Navigate to another screen within the app (e.g., from the main screen to the profile screen).  2. Trigger a data request (e.g., refresh the page or load a list of items).  3. Fill out and submit a form within the app.  4. Measure the response time for each of these actions | Screen transition: not more than 2 seconds  Data request: not more than 2-3 seconds  Form submission: not more than 1-2 seconds | Pass |  | High |
| **12.3** | **Endurance Testing Under Prolonged Load** | Assess application stability under prolonged load to identify potential performance issues | The app is running; the device is charged to at least 80% or connected to a power source | 1. Launch the application and leave it running in the background for 4 hours.  2. Every 30 minutes, return to the app and perform basic user actions (such as screen navigation, data loading).  3. Monitor app performance, noting any lags, crashes, or delays in response time.  4. At the end of the test, repeat the initial actions and verify if response times have changed | The app remains stable, with no significant delays or errors even after hours of continuous use; response time remains consistent | Pass |  | Medium |
| **12.4** | **Verification of Efficient Memory Usage** | Verify that the application manages memory effectively without causing excessive consumption or memory leaks that could impact device performance | The application is installed on a device with monitoring tools (e.g., Android Studio Profiler or Xcode Instruments)  The device has been recently restarted to clear memory usage | 1. Open the application and navigate through multiple screens (e.g., home, settings, profile).  2. Perform repetitive actions, such as loading data, switching screens, and interacting with UI elements.  3. Use a memory monitoring tool to observe memory consumption over time, particularly for any noticeable increase during the test | The application maintains a stable memory footprint without significant increases over time. No memory leaks or crashes are detected. | Pass |  | High |
| **12.5** | **Verification of CPU Usage Efficiency** | Check that the application’s CPU usage is within acceptable limits and does not spike unnecessarily, which could impact the device's performance | The application is installed on a device with a CPU monitoring tool (e.g., Android Studio Profiler or Xcode Instruments) | 1. Launch the application and perform common tasks, such as browsing content, refreshing feeds, or submitting forms.  2. Monitor CPU usage while performing these actions, noting any spikes or sustained high usage.  3. Let the application run in the background and observe CPU usage to ensure minimal resource drain | CPU usage remains within a reasonable range during normal use and minimizes usage in the background. No sustained spikes in CPU utilization are observed | Pass |  | High |
| **12.6** | **Verification of Battery Usage Efficiency** | Verify that the application does not excessively drain the device’s battery, especially when running in the background or under extended use | The application is installed on a fully charged device.  Access to battery monitoring tools or settings on the device to track app-related consumption | 1. Fully charge the device’s battery, then launch the application and use it for 30-60 minutes, performing standard tasks (e.g., navigating screens, loading data).  2. Check the battery usage statistics on the device to monitor the application’s impact.  3. Leave the application running in the background for an additional 30-60 minutes and observe any changes in battery consumption | The application consumes battery within an acceptable limit 10-15% and does not cause excessive drain, either during active usage or when running in the background | Pass |  | Medium |
| **12.7** | **Load Testing Under High User Volume** | Verify the application’s ability to handle a large number of concurrent users without performance degradation | A test environment that can simulate high user load (e.g., JMeter or LoadRunner).  Access to a staging server/environment to handle high-traffic scenarios | 1. Set up a load testing tool to simulate a high number of concurrent users (e.g., 1,000+ users).  2. Configure test scenarios that include common actions such as logging in, browsing, and data submission.  3. Gradually increase the load to observe the application’s performance metrics, including response time, error rates, and server resource usage | The application maintains acceptable response times and does not exhibit any significant performance issues, such as crashes or error messages, even as user load increases | Pass |  | High |
| **12.8** | **Stress Testing for Peak Load Durability** | Verify the application’s stability and behavior under extreme load conditions to determine the breaking point | A load testing tool (e.g., JMeter or LoadRunner).  A dedicated test environment to simulate peak loads without affecting production | 1. Gradually increase the number of simulated users to above-average levels, targeting up to 150% or 200% of expected maximum usage.  2. Observe application performance metrics, including response times, error rates, and server load.  3. Document any system failures, bottlenecks, or significant performance drops as the load reaches peak levels | The application remains stable and functional within expected load limits but may exhibit reduced performance or controlled failures at peak load levels. Clear insights on the application’s load tolerance are identified | Pass |  | Medium |

### **13. Security Testing**

| **ID** | **Title** | **Description** | **Precondition** | **Test steps** | **Expected result** | **Status** | **Notes** | **Priority** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **13.1** | **Check Data Transmission over Secure Channels** | Verify whether all data transmitted from the mobile application is done over secure channels (HTTPS) to ensure user data security | The mobile application is installed on the test device.  A traffic interception tool (Burp Suite or Fiddler) is set up and running.  The mobile device is connected to a Wi-Fi network with the computer running the interception tool.  The security certificate of the interception tool is installed on the mobile device (for HTTPS) | 1. Launch Burp Suite/Fiddler on the computer.  2. Configure the proxy server on the mobile device, specifying the computer's IP address and the port where the tool is running.  3. Install the security certificate of the interception tool on the mobile device (if necessary).  4. Open the mobile application on the test device.  5. Perform actions that involve data transmission (e.g., logging in, submitting a form).  5. Observe the traffic in Burp Suite/Fiddler | All requests sent from the mobile application should have the https:// protocol.  No requests should be made via http://.  Sensitive data (e.g., passwords) should be encrypted and not appear in plain text in the intercepted traffic | Pass |  | High |
| **13.2** | **Check Storage of Confidential Data** | Verify how the mobile app stores confidential data (e.g., passwords) to ensure that it is stored in an encrypted format or not stored locally at all | The mobile application is installed  The user has an account with confidential data | 1. Log in using credentials that include confidential information (password).  2. Log out of the application and close it.  3. Use a memory reading tool (e.g., APKTool or similar) to check local data storage | Confidential data (passwords, tokens) should not be stored in plain text in the device's memory.  If data is stored, it should be encrypted | Pass |  | High |
| **13.3** | **Check Access Restrictions and Authorization** | Verify whether the application correctly restricts access to certain features for users without proper authorization | The mobile application is installed.  The user does not have administrative access rights | 1. Attempt to access features that require elevated access (e.g., administration, changing settings).  2. Record the access results | The application should deny access attempts or prompt for additional authorization (e.g., password or confirmation) | Pass |  | High |
| **13.4** | **Check Abuse Prevention through Rate Limiting and Account Lockout** | Checks if the application includes protections against brute force attacks by rate-limiting login attempts or locking accounts after multiple failed attempts | The application has a login feature.  A testing account is available | 1. Attempt to log in with incorrect credentials multiple times in succession.  2. Observe the application’s response after repeated failed attempts | The application should enforce rate limiting, delaying repeated login attempts.  If attempts exceed a certain threshold, the application should temporarily lock the account or require additional verification (e.g., CAPTCHA) | Pass |  | High |

**14. Сompatibility Testing**

| **Test cases for a mobile application, covering testing on various devices and different OS versions can be found at the following link:** [**https://github.com/OleksandraSolodka/Prometheus-Internship-QAManual-Program/blob/main/Skybot%20Mobile%20Application/OSolodka\_CompatibilityTestingSMAVersion2.0.docx**](https://github.com/OleksandraSolodka/Prometheus-Internship-QAManual-Program/blob/main/Skybot%20Mobile%20Application/OSolodka_CompatibilityTestingSMAVersion2.0.docx) |
| --- |

**15. Fault Tolerance Testing**

| **ID** | **Title** | **Description** | **Precondition** | **Test steps** | **Expected result** | **Status** | **Notes** | **Priority** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **15.1** | **Resilience to Invalid Data Input in Forms** | Verify if the application correctly handles invalid data inputs and avoids crashes | The application is installed.  Access to a data input screen (e.g., registration or login form) | 1. Open the data input form (e.g., registration).  2. Enter invalid data such as:  Blank fields.  Incorrect email format.  Invalid characters in text fields (e.g., special characters or SQL injections).  3. Submit the form. | The app displays an error message indicating invalid data.  The app does not crash | Pass |  | High |
| **15.2** | **Application Resilience During Server Failure** | Verify how the application behaves if the server fails during an ongoing operation (e.g., sending a request) | Network access or ability to disconnect network connection.  The application is installed on the test device. | 1. Launch the app and perform an action requiring server interaction (e.g., updating profile or sending a message).  2. Disconnect from the internet or simulate a server failure during the request.  3. Observe the app’s behavior | The app shows an appropriate error message indicating the connection failure.  The app does not crash and allows the user to retry the action after reconnection | Pass |  | Medium |
| **15.3** | **Recovery after Server Downtime During Data Load** | Verify if the application can recover and retry loading data after a temporary server failure | The application is installed on the test device with server access.  Temporary disconnection from the server is possible | 1. Open the app and navigate to a screen requiring data loading from the server (e.g., messages or user info).  2. Temporarily disconnect the server or network connection.  3. Wait for a while and then reconnect | The app notifies the user about data loading issues during disconnection.  The app retries and successfully loads data after reconnection | Pass |  | High |
| **15.4** | **App Stability with Prolonged Server Response Times** | Verify if the app remains stable when there is a prolonged response time from the server | The application is installed on the test device with access to the server.  Server configured for delayed response times | 1. Initiate a server-dependent action (e.g., loading a large amount of data).  2. Introduce a server delay to slow down the response.  3. Observe the app’s behavior during the wait time | The app remains stable without crashing.  Once the response is received, the app completes the operation smoothly | Pass |  | Medium |

**16. Accessibility testing**

| **ID** | **Title** | **Description** | **Precondition** | **Test steps** | **Expected result** | **Status** | **Notes** | **Priority** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **16.1** | **Accessibility Version Check** | Ensure that the application has an accessible version for users with disabilities | The application is installed on the device | 1. Open the application.  2. Enable accessibility mode on the device (e.g., TalkBack on Android).  3. Review all interface elements. | The application should have an accessible version that complies with accessibility standards (e.g., WCAG) | Pass |  | High |
| **16.2** | **UI Element Label Check** | Verify that the application uses correct labels for all user interface elements | The application is installed on the device | 1. Open the application.  2. Use an accessibility testing tool (e.g., Accessibility Scanner or Axe).  3. Review all user interface elements | All user interface elements should have clear and accurate labels explaining their purpose | Fail | Interface elements don’t have clear and accurate labels explaining their purpose | High |
| **16.3** | **Color Contrast Test** | Check if the application has adequate color contrast for users with visual impairments | The application is installed on the device | 1. Open the application.  2. Use a color contrast checking tool (e.g., Color Contrast Analyzer).  3. Check the contrast between text and background on all screens. | The color contrast should meet WCAG standards (minimum 4.5:1 for normal text) | Fail | The color contrast doesn’t meet WCAG standards | High |
| **16.4** | **Alternative Text Check for Images** | Verify that all images in the application have alternative text | The application is installed on the device | 1. Open the application.  2. Browse through all screens containing images.  3. Check for the presence of alternative text for each image | All images should have appropriate alternative text that describes their content | Fail | Images don’t have alternative text describes content | High |