Assignment:

Test Plan

Image Generated by Dalle-E 2 from chat GPT

Prepared for

Nomaan Beg, Mirza

Prepared by Group 7

Valentine Jingwa, Jean-Pierre Nde-Forgwang, Jonah Scott, Nico Gonzalez

Security System, Cohort A

School of Advance Digital Technology

SAIT

2 March 2024

Index

Overall Testing Schedule:

This is a master schedule that includes all test case documentation, execution, and reporting schedules across all development phases.

Phase 1: Test Planning and Documentation

March 3 - March 10, 2024

- Objectives: Finalize the test plan, identify and understand all test cases, objectives, as well as outlining the scope and limitations.
- Activities: Team meetings to finalize our strategy, start documenting test environments and data requirements.

Phase 2: Test Case Review

March 11 - March 18, 2024

- Objectives: Complete a detailed list on each test case describing each functionality and what should be the outcome, and possible scenarios when testing.
- Activities: Review test cases for accuracy and completeness and adjust if necessary.

Phase 3: Test Environment Setup

March 19 - March 21, 2024

- Objective: Prepare all test environments, and gather all required tools and application access is in place for all tests.
- Activities: Set up and verify all testing environments are correct with the right data and tools available.

Phase 4: Test Executions

March 22 - April 4, 2024

- Objective: Execute all planned test cases in this document, and record all issues or successes encountered.
- Activities: Run tests according to the execution schedule (By page) and log all results and create documentation for each test case on any defects or issues found.

Phase 5: Test Result Analysis

April 5 - April 10, 2024

- Objective: Analyze test results, prioritize the most critical defects and issues that will hinder our applications goals.
- Activities: Review each test outcome and report all issues and successes to a shared group document along with their impact, for group analysis.

Phase 6: Retesting

April 11 - April 15, 2024

- Objective: Verify all critical and moderate defects have been successfully mitigated and addressed in initial testing, and start retesting to determine if new issues arise.
- Activities: Conduct retesting of all fixed defects to determine if new issues have been introduced into the system.

Phase 7: Final Review, Documentation and Submission

April 16 - April 19, 2024

- Objective: Finalize all testing documentation, include a summary of the testing activity outcomes, and recommend future improvements on the system.
- Activities: Compile all testing reports into a document, and review each testing activity with the team, instructor, and stakeholders for submission, and submit our final test plan and application.

Home page

CID: 1 - Title: Homepage Navigation Button

- *Description:* Test case to verify the functionality and responsiveness of the navigation buttons on the Home Page

- *Development Phase:* This test case falls under the Integration Testing phase, where individual components are combined and tested as a group to ensure their correct interactions within the application.

- *Test Goals and Measurement:* Ensure that navigation buttons are functional and responsive and that data retrieval/display mechanisms work as expected. All navigation buttons must respond within 2 seconds, correctly transition between views, and accurately retrieve/display data for selected dates without errors.

- Test Data/ Tools: Pre-populated data for various dates to test data retrieval and display functionality. Mobile testing framework Network simulation tools for testing under different network conditions.
- Test Startup/Closedown/Reset Procedure:
    - *Startup:* Ensure the mobile app is installed on the test device or emulator. Start the app and log in if necessary.
    - *Closedown:* Close the app and clear it from recent applications to ensure a fresh start for each test.
    - *Reset:* Clear app data or reinstall the app to reset any saved states or preferences to default.

- *Execution Steps:*

- Step: Navigate to the Home Page.
  Action: Open the app and authenticate.
  Expected Result: The Home Page loads successfully with navigation buttons visible.

- *Execution Metrics:*
Placeholder first attempt to execute this test case date/time: [To be filled out during testing]
Placeholder number of attempts to successfully execute this test case: [To be filled out during testing]
Placeholder list of software defects discovered by this test case: [To be filled out during testing]
Placeholder is this test case executed successfully? (yes/no): [To be filled out during testing]

CID: 2 - Title: Calendar Functionality

- *Description:* This test case assesses the calendar's functionality within the app, focusing on the ability to select dates, the accuracy of displaying data corresponding to selected dates, and the overall responsiveness and usability of the calendar feature.

- *Development Phase:* This test falls under the System Testing phase, as it evaluates the calendar feature's integration within the app, ensuring it operates correctly within the entire system.

- *Test Goals and Measurement:* Verify that the calendar feature is fully operational, allowing users to select dates, and ensuring the app accurately displays data for the selected dates. Additionally, assess the feature's usability and responsiveness. Success is measured by the calendar's ability to update views promptly upon date selection, accurately reflect data for selected dates, and maintain high usability standards with no user interface lag.

- Test Data/ Tools: Create a dataset with entries for multiple dates across several months to ensure the calendar can retrieve and display data across various scenarios. Test app functionality with an Android emulator tool.
- Test Startup/Closedown/Reset Procedure:

- *Startup:* Ensure the app is installed on the device or emulator. Start the app, navigate to the calendar feature, and ensure it is in its default state.
- *Closedown:* Close the app and clear it from the device or emulator's recent applications list to ensure no residual data affects the test.
- *Reset:* Clear the app's data or reinstall it to reset the calendar to its default state, ensuring a consistent starting point for each test iteration.

- *Execution Steps:*

- Step: Open the calendar feature within the app.
  Action: Launch the app and navigate to the page or section where the calendar is accessible.
  Expected Result: The calendar loads correctly and displays the current month.

- Step: Select a specific date with known data entries.
  Action: Tap on a date for which test data has been created.
  Expected Result: The app responds by highlighting the selected date and displaying data or events associated with it.

- Step: Navigate between months.
  Action: Use the calendar's navigation to move to the next and previous months.
  Expected Result: The calendar smoothly transitions between months, and data corresponding to each month is correctly displayed if applicable.

- Step: Select a date with no data entries.
  Action: Tap on a date that does not have any data associated with it.
  Expected Result: The app correctly indicates that there are no entries/data for the selected date, without crashing or displaying

incorrect information.

- Step: Assess calendar responsiveness and usability.
  Action: Interact with the calendar by selecting various dates, navigating between months, and observing the interface's responsiveness.
  Expected Result: The calendar feature is responsive, with no significant lag or usability issues, enhancing user experience.

- *Execution Metrics:*
Placeholder first attempt to execute this test case date/time: [To be filled out during testing]
Placeholder number of attempts to successfully execute this test case: [To be filled out during testing]
Placeholder list of software defects discovered by this test case: [To be filled out during testing]
Placeholder is this test case executed successfully? (yes/no): [To be filled out during testing]

CID: 3 - Title: Calendar Back Button

- *Description:* This test case verifies the functionality of the back button within the calendar feature of the app. It focuses on the button's ability to return the user to the previous Month/Year from the calendar view without losing any unsaved data or state.

- *Development Phase:* This test falls under the User Interface Testing phase. This phase focuses on the graphical elements and user interaction with the app, ensuring that navigation elements like the back button work as intended.

- *Test Goals and Measurement:* Ensure the calendar's back button correctly navigates the user back to the previously accessed screen or state, maintaining the app's navigational integrity and user experience. The success of this test is measured by the back button's responsiveness and its ability to return the user to the correct previous state or screen without errors or data loss.

- Test Data/ Tools: No specific data is required for this test, but having various states within the calendar to return to can help in thorough testing. Test app functionality with an Android emulator tool.
- Test Startup/Closedown/Reset Procedure:
    - *Startup:* Launch the app and navigate to the calendar feature, ensuring it is loaded properly.
    - *Closedown:* Exit the app and remove it from the recent applications list to ensure a fresh state for each test run.
    - *Reset:* If necessary, restart the device or emulator to clear all temporary states and cache, ensuring the app starts from a clean state for each test.

- *Execution Steps:*

- Step: Access the calendar feature within the app.
  Action: Open the app, authenticate, and navigate to the calendar.
  Expected Result: The calendar is displayed correctly.

- Step: Navigate to a specific date or month within the calendar.
  Action: Select a different month or date from the current view.
  Expected Result: The calendar updates to display the selected month or date.

- Step: Use the back button from the calendar view.
  Action: Press the back button on the device.
  Expected Result: The app returns to the previous screen or state before accessing the calendar.

- Step: Verify state retention after using the back button.
  Action: After returning to the previous screen, navigate back to the calendar to check if the last viewed state (month or date) is retained.
  Expected Result: The calendar displays the last state viewed before the back button was used, indicating state retention.

- *Execution Metrics:*
Placeholder first attempt to execute this test case date/time: [To be filled out during testing]
Placeholder number of attempts to successfully execute this test case: [To be filled out during testing]
Placeholder list of software defects discovered by this test case: [To be filled out during testing]
Placeholder is this test case executed successfully? (yes/no): [To be filled out during testing]

CID: 4 - Title: Calendar Forward Button

- *Description: Description:* This test case verifies the functionality of the forward button within the calendar feature of the app. It focuses on the button's ability to send the user to the next Month/Year from the calendar view without losing any unsaved data or state.

- *Development Phase:* This test falls under the User Interface Testing phase. This phase focuses on the graphical elements and user interaction with the app, ensuring that navigation elements like the forward button work as intended.

- *Test Goals and Measurement:* Ensure the calendar's forward button correctly navigates the user to the next accessed screen or state, maintaining the app's navigational integrity and user experience. The success of this test is measured by the forward button's responsiveness and its ability to send the user to the correct state or screen without errors or data loss.

- Test Data/ Tools: No specific data is required for this test, but having various states within the calendar to return to can help in thorough testing. Test app functionality with an Android emulator tool.
- Test Startup/Closedown/Reset Procedure:
    - *Startup:* Launch the app and navigate to the calendar feature, ensuring it is loaded properly.
    - *Closedown:* Exit the app and remove it from the recent applications list to ensure a fresh state for each test run.

- *Reset:* If necessary, restart the device or emulator to clear all temporary states and cache, ensuring the app starts from a clean state for each test.

- Execution Steps:

- Step: Access the calendar feature within the app.
  Action: Open the app, authenticate, and navigate to the calendar.
  Expected Result: The calendar is displayed correctly.

- Step: Navigate to a specific date or month within the calendar.
  Action: Select a different month or date from the current view.
  Expected Result: The calendar updates to display the selected month or date.

- Step: Use the forward button from the calendar view.
  Action: Press the forward button on the device.
  Expected Result: The app sends you to the next screen or state before accessing the calendar.

- Step: Verify state retention after using the forward button.
  Action: After returning to the next screen, navigate back to the calendar to check if the last viewed state (month or date) is retained.
  Expected Result: The calendar displays the last state viewed before the back button was used, indicating state retention.

- Execution Metrics:
Placeholder first attempt to execute this test case date/time: [To be filled out during testing]
Placeholder number of attempts to successfully execute this test case: [To be filled out during testing]
Placeholder list of software defects discovered by this test case: [To be filled out during testing]
Placeholder is this test case executed successfully? (yes/no): [To be filled out during testing]

CID: 5 - Title: Calendar Text Color

- *Description:* This test case evaluates the visibility and readability of text within the calendar feature of the app, focusing on text color contrast against the calendar's background.

- *Development Phase:* This test is part of the Accessibility Testing phase, aiming to ensure that the app's calendar is easily readable and accessible to all users, including those with visual impairments.

- *Test Goals and Measurement:* Confirm that the text color in the calendar provides sufficient contrast against the background for easy readability under various lighting conditions. Success is measured by the text meeting or exceeding the minimum contrast ratios defined by WCAG (Web Content Accessibility Guidelines) for text visibility.

- Test Data/ Tools: No Test Data required. Test app functionality with an Android emulator tool.
- Test Startup/Closedown/Reset Procedure:
    - *Startup:* Open the app and navigate to the calendar feature.
    - *Closedown:* Close the app after testing.
    - *Reset:* Not required for this test.

- *Execution Steps:*

- Step: Open the calendar within the app.
  Action: Launch the app and navigate to the calendar feature.
  Expected Result: The calendar is displayed with all text visible.

- Step: Examine text color contrast.
  Action: Use a contrast analyzer tool to measure the contrast ratio of the text against the calendar's background.
  Expected Result: The contrast ratio meets or exceeds the minimum requirements for text visibility according to WCAG guidelines.

- *Execution Metrics:*
Placeholder first attempt to execute this test case date/time: [To be filled out during testing]
Placeholder number of attempts to successfully execute this test case: [To be filled out during testing]
Placeholder list of software defects discovered by this test case: [To be filled out during testing]
Placeholder is this test case executed successfully? (yes/no): [To be filled out during testing]

CID: 6 - Title: Data Retrieval By Date

- *Description:* This test case aims to verify the app's ability to accurately retrieve and display data associated with specific dates selected in the calendar.

- *Development Phase:* Falls under the Functional Testing phase, specifically focusing on verifying the app's functionality to ensure it works as expected for data retrieval based on user-selected dates.

- *Test Goals and Measurement:* Ensure accurate data retrieval and display for any date selected within the calendar. Every selection of a date should accurately display data for that date, with no discrepancies or errors.

- Test Data/ Tools: Pre-populated data for various dates within the app's database. Database management tools and test app functionality with an Android emulator tool.
- Test Startup/Closedown/Reset Procedure:
    - *Startup:* Launch the app, navigate to the calendar feature, and ensure it loads correctly.
    - *Closedown:* Close the app after each test run.
    - *Reset:* Clear app data or reinstall the app if necessary to ensure a clean state for each test.

- *Execution Steps:*
- Step: Navigate to the calendar feature and select a date with known data. Action: Choose a date for which data has been pre-populated.

Expected Result: The app displays the data associated with the selected date accurately.

- Step: Verify the accuracy of displayed data.
  Action: Compare the displayed data against the expected data set for that date.
  Expected Result: Displayed data matches the expected data set without errors.

- Step: Test data retrieval for multiple dates.
  Action: Repeat the data selection process for various dates across different months and years.
  Expected Result: For each selected date, the app accurately retrieves and displays the corresponding data.

- *Execution Metrics:*
Placeholder first attempt to execute this test case date/time: [To be filled out during testing]
Placeholder number of attempts to successfully execute this test case: [To be filled out during testing]
Placeholder list of software defects discovered by this test case: [To be filled out during testing]
Placeholder is this test case executed successfully? (yes/no): [To be filled out during testing]

CID: 7 - Title: Data Display

- *Description:* This test case evaluates the app's capability to correctly and efficiently display data to the user after retrieval, focusing on the accuracy, format, and readability of the displayed data.

- *Development Phase:* This falls under the User Acceptance Testing (UAT) phase. UAT is crucial for ensuring that the app meets the end-users' requirements and expectations in real-world scenarios, particularly regarding how data is presented to them.

- *Test Goals and Measurement:* To confirm that the app displays data accurately

and in a user-friendly format after performing data retrieval actions. The data displayed should match the expected results with 100% accuracy, follow the predefined format specifications, and maintain high readability and user engagement standards.

- Test Data/ Tools: A diverse set of data entries that covers all types of data the app is expected to display. I.e Vitals/Medication/Nutrition/Others
- Test Startup/Closedown/Reset Procedure:
    - *Startup:* Begin with the app launched, and navigate the Home page.
    - *Closedown:* Properly close the app and clear it from recent apps to ensure no data caching affects subsequent tests.
    - *Reset:* Clear app data or reinstall the app between tests if necessary to return to a clean, default state.

- *Execution Steps:*
    - Step: Navigate to the Home page.
      Action: Access a part of the app known to retrieve and display data, such as a user's health log or calendar events.
      Expected Result: The selected section loads without errors, displaying the retrieved data.

    - Step: Verify the accuracy of the displayed data.
      Action: Compare the data shown in the app with the expected data set for accuracy.
      Expected Result: The data displayed in the app matches the expected data accurately.

    - Step: Assess the format and readability of the displayed data.
      Action: Evaluate if the data is presented in a clear, understandable format and is easy to read.
      Expected Result: The format of the displayed data is consistent with design specifications, and readability standards are met.

- Step: Test the responsiveness of the data display to user interactions.
  Action: Interact with the data display features, such as scrolling, zooming, or selecting different data categories.
  Expected Result: The app responds promptly to interactions, with no lag or misalignment in the data display.

- *Execution Metrics:*
Placeholder first attempt to execute this test case date/time: [To be filled out during testing]
Placeholder number of attempts to successfully execute this test case: [To be filled out during testing]
Placeholder list of software defects discovered by this test case: [To be filled out during testing]
Placeholder is this test case executed successfully? (yes/no): [To be filled out during testing]

CID: 8 - Title: Recent Data Display By Entry

- *Description:* This test case aims to verify the app's functionality to correctly display the most recent data entries to the user, focusing on the accuracy of the data shown, the order of the entries, and the system's ability to update this view as new data is entered.

- *Development Phase:* This test is best situated in the Functional Testing phase, specifically under Regression Testing to ensure that as new features are added or updates are made, the functionality to display recent data entries correctly remains intact.

- *Test Goals and Measurement:* Ensure the app accurately displays the most recent data entries in the correct order and updates this display in real-time as new entries are made. Success is measured by the app's ability to display the latest entries at the top of the retrieve data section on the Home page, with updates reflecting immediately after new data is entered.

- Test Data/ Tools: Create a series of data entries with timestamps to simulate real user entries. Ensure there's a mix of older and newer

entries to test the sorting and display functionality. Test app functionality with an Android emulator tool.

- Test Startup/Closedown/Reset Procedure:
    - *Startup:* Launch the app, navigate to the Home page, under the data retrieval section, and ensure no entries are present.
    - *Closedown:* Close the app and clear it from the recent apps list to ensure a fresh environment for each test iteration.
    - *Reset:* Clear the app's data or reinstall the app if necessary to reset the environment to its default state before testing begins.

- *Execution Steps:*

- Step: Verify the initial state of the data display.
  Action: Open the app and view the section where recent entries should be displayed.
  Expected Result: The most recent entries are displayed at the top, in descending order by date/time.

- Step: Add a new data entry.
  Action: Enter new data into the app, ensuring it has a current timestamp.
  Expected Result: Upon submission, the new entry should immediately appear at the top of the recent entries display.

- Step: Add multiple new data entries.
  Action: Repeat the data entry process with several new entries, each with its unique timestamp.
  Expected Result: Each new entry appears at the top of the list, pushing older entries down, and maintaining the correct order by date/time.

- Step: Refresh or navigate away and back to the recent entries displayed.
  Action: After adding new entries, refresh the display or navigate to another section of the app and then return to the recent entries display.

Expected Result: The display still accurately shows the most recent entries in the correct order, indicating the app correctly saves and retrieves the data.

- *Execution Metrics:*
Placeholder first attempt to execute this test case date/time: [To be filled out during testing]
Placeholder number of attempts to successfully execute this test case: [To be filled out during testing]
Placeholder list of software defects discovered by this test case: [To be filled out during testing]
Placeholder is this test case executed successfully? (yes/no): [To be filled out during testing]

CID: 9 - Title: Data Display Saved Functionality

- *Description:* Evaluates the app's ability to save data input by users and subsequently display the saved data accurately within the app, ensuring data persistence across sessions.

- *Development Phase:* This test case is part of the Integration Testing phase, focusing on the app's capacity to integrate data saving and retrieval processes smoothly and ensuring data persistence.

- *Test Goals and Measurement:* Confirm that data entered by users is saved correctly and displayed accurately when revisited, even after app restarts or logouts. Success is measured by the app's ability to display the previously saved data without any loss or alteration, maintaining 100% data integrity.

- Test Data/ Tools: Sample data inputs that cover all data types the app allows users to save. Database management tools and test app functionality with an Android emulator tool.
- Test Startup/Closedown/Reset Procedure:
    - *Startup:* Launch the app and navigate to the feature where data saving functionality is to be tested.
    - *Closedown:* Properly exit the app and, if necessary, log out or restart the device to simulate a new session.

- *Reset:* Clear app data or reinstall the app between test runs if needed to ensure starting from a clean slate.

- *Execution Steps:*

- Step: Input and save data using the app's functionality.
  Action: Enter data into the app as a user would and use the provided mechanism to save it.
  Expected Result: The app confirms that data is saved.

- Step: Close and reopen the app or log out and back in.
  Action: Simulate leaving the session by closing the app or logging out, then returning.
  Expected Result: Upon return, the user is able to access the previously saved data exactly as entered.

- Step: Verify data integrity and accuracy.
  Action: Compare the displayed saved data against the original input for accuracy.
  Expected Result: The saved data displayed matches the original input perfectly, with no loss or alteration.

- *Execution Metrics:*
Placeholder first attempt to execute this test case date/time: [To be filled out during testing]
Placeholder number of attempts to successfully execute this test case: [To be filled out during testing]
Placeholder list of software defects discovered by this test case: [To be filled out during testing]
Placeholder is this test case executed successfully? (yes/no): [To be filled out during testing]

CID: 10 - Title: Data Display Scrollability

- *Description:* Tests the scroll functionality within the app's data display sections, ensuring users can smoothly scroll through data without issues.

- *Development Phase:* This test is conducted during the User Interface Testing phase, focusing on the usability aspect of the app, particularly the smoothness and responsiveness of scrolling actions.

- *Test Goals and Measurement:* Ensure the app provides a smooth and responsive scrolling experience in data display sections, without lag, stuttering, or crashes. Scrolling actions are smooth and responsive across different devices and data volumes, with no reported issues.

- Test Data/ Tools: Large datasets to fill the display sections, ensuring there's enough content to scroll through. Test app functionality with an Android emulator tool.
- Test Startup/Closedown/Reset Procedure:
    - *Startup:* Launch the app and navigate to the Home screen, to the data section intended for scrolling.
    - *Closedown:* Close the app after testing to reset its state.
    - *Reset:* Clear the app's cache or reinstall if necessary to ensure default settings are restored for each test run.

- *Execution Steps:*

- Step: Navigate to Data Display Section
  Action: Launch the app and navigate from the Home screen to the section where data is displayed and requires scrolling.
  Expected Result: The data display section is successfully loaded with visible data ready for interaction.

- Step: Initial Scroll Test
  Action: Perform a quick swipe up and down to initiate scrolling through the displayed data.
  Expected Result: The app responds promptly to the swipe actions, allowing for smooth scrolling without noticeable lag or stuttering.

- Step: Extended Scroll Test
  Action: Continuously scroll through the data display section to the end of the data set and back to the top.
  Expected Result: Scrolling is smooth and consistent throughout, with no performance degradation or crashes.

- Step: Scroll with Varied Speeds
  Action: Scroll through the data display section at different speeds, including slow, medium, and fast swipes.
  Expected Result: The app accurately registers different scrolling speeds and adjusts the scroll velocity accordingly, without any issues.

- *Execution Metrics:*
Placeholder first attempt to execute this test case date/time: [To be filled out during testing]
Placeholder number of attempts to successfully execute this test case: [To be filled out during testing]
Placeholder list of software defects discovered by this test case: [To be filled out during testing]
Placeholder is this test case executed successfully? (yes/no): [To be filled out during testing]

Viewing Page

CID: 11 - Title: Viewing Page Navigation Button

- *Description:* This use case focuses on evaluating the app's navigation button functionality on the viewing page. The goal is to ensure seamless navigation for users as they interact with the app, specifically assessing the responsiveness and accuracy of navigation buttons.

- *Development Phase:* This falls under the User Acceptance Testing (UAT) phase. UAT is essential to guarantee that the app meets end-users' requirements and expectations in real-world scenarios, particularly regarding navigation features

- *Test Goals and Measurement:* To confirm that the app's navigation buttons on

the viewing page function accurately, respond promptly to user interactions, and provide an intuitive and user-friendly experience. Measurement will be based on the successful execution of navigation actions and the absence of navigation-related defects.

- Test Data/ Tools: A set of diverse data entries that covers all types of information the app displays on the viewing page. Additionally, tools for tracking navigation responsiveness and accuracy.
- Test Startup/Closedown/Reset Procedure:
    - *Startup:* Begin with the app launched, and navigate to the viewing page.
    - *Closedown:* Properly close the app and clear it from recent apps to ensure no data caching affects subsequent tests.
    - *Reset:* Clear app data or reinstall the app between tests if necessary to return to a clean, default state.

- *Execution Steps:*

1. Navigate to the Viewing Page
    - Action: Open the app and navigate to the viewing page.
    - Expected Result: The viewing page loads without errors, displaying the relevant data.
2. Locate Navigation Buttons
    - Action: Identify and locate the navigation buttons on the viewing page.
    - Expected Result: Clear visibility of navigation buttons, placed intuitively for user access.
3. Test Navigation Button Functionality
    - Action: Interact with each navigation button (e.g., next, previous, back) to navigate between different sections or items on the viewing page.
    - Expected Result: The app responds promptly and accurately to each navigation button press, transitioning to the intended section or item.

4. Verify Accuracy of Navigation
    - Action: Navigate through different sections using the buttons and verify that the displayed content corresponds to the expected data for each section.
    - Expected Result: Accurate navigation, ensuring that the displayed data matches the expected information for the selected section.
5. Assess User Interface Responsiveness
    - Action: Test the responsiveness of navigation buttons to user interactions, such as tapping or swiping.
    - Expected Result: Smooth and responsive interactions without delays or glitches.

- *Execution Metrics:*
Placeholder first attempt to execute this test case date/time: [To be filled out during testing]
Placeholder number of attempts to successfully execute this test case: [To be filled out during testing]
Placeholder list of software defects discovered by this test case: [To be filled out during testing]
Placeholder is this test case executed successfully? (yes/no): [To be filled out during testing]

CID: 12 - Title: Vital Sub-Category Displayed

- *Description:*This use case aims to evaluate the app's ability to accurately display vital sub-category information, ensuring that the data is presented correctly and meets the defined standards. The focus is on the specific sub-categories related to vital information within the app.

- *Development Phase:* This falls under the User Acceptance Testing (UAT) phase. UAT is vital for confirming that the app meets the end-users' requirements and expectations regarding the display of vital sub-categories.

- *Test Goals and Measurement:* The primary goal is to verify that the app accurately displays vital sub-category information. Measurement will be based on the accuracy of the displayed data, adherence to predefined format

specifications, and the overall user experience in accessing vital sub-category details.

- Test Data/ Tools: A set of diverse data entries representing various vital sub-categories that the app is expected to display. Additionally, tools for tracking data accuracy and format compliance.
- Test Startup/Closedown/Reset Procedure:
    - *Startup:* Begin with the app launched, and navigate to the section displaying vital sub-category information.
    - *Closedown:* Properly close the app and clear it from recent apps to ensure no data caching affects subsequent tests.
    - *Reset:* Clear app data or reinstall the app between tests if necessary to return to a clean, default state.

- *Execution Steps:*

1. Navigate to the Vital Sub-Category Section
    - Action: Open the app and navigate to the section displaying vital sub-category information.
    - Expected Result: The section loads without errors, presenting the vital sub-categories available for viewing.
2. Locate and Identify Vital Sub-Categories
    - Action: Identify and locate the different vital sub-categories displayed in the section.
    - Expected Result: Clear visibility of vital sub-categories with proper labeling and organization.
3. Verify Accuracy of Displayed Data
    - Action: Check each vital sub-category for the accuracy of the displayed data.
    - Expected Result: The app accurately presents vital information, matching the expected data set for each sub-category.
4. Assess Format and Readability
    - Action: Evaluate if the vital sub-category data is presented in a clear, understandable format and is easy to read.

- Expected Result: The format of the displayed data is consistent with design specifications, and readability standards are met.

5. Test User Interaction with Vital Sub-Categories
    - Action: Interact with the displayed vital sub-categories, such as tapping or selecting different sub-categories.
    - Expected Result: The app responds promptly to interactions, allowing users to access detailed information for each vital sub-category.

- *Execution Metrics:*
Placeholder first attempt to execute this test case date/time: [To be filled out during testing]
Placeholder number of attempts to successfully execute this test case: [To be filled out during testing]
Placeholder list of software defects discovered by this test case: [To be filled out during testing]
Placeholder is this test case executed successfully? (yes/no): [To be filled out during testing]

CID: 13 - Title: Medication Subcategory Displayed

- *Description:* This use case focuses on evaluating the app's ability to accurately display medication subcategory information. The objective is to ensure that the app presents medication-related data correctly and adheres to the specified standards.

- *Development Phase:* This falls under the User Acceptance Testing (UAT) phase. UAT is crucial for confirming that the app meets end-users' requirements and expectations, specifically in displaying medication subcategory details.

- *Test Goals and Measurement:* The primary goal is to verify that the app accurately displays medication subcategory information. Measurement will be based on the accuracy of the displayed data, adherence to predefined format specifications, and the overall user experience in accessing medication subcategory details.

- Test Data/ Tools: A set of diverse data entries representing various medication sub-categories that the app is expected to display. Additionally, tools for tracking data accuracy and format compliance.
- Test Startup/Closedown/Reset Procedure:

    - *Startup:* Begin with the app launched, and navigate to the section displaying medication subcategory information.
    - *Closedown:* Properly close the app and clear it from recent apps to ensure no data caching affects subsequent tests.
    - *Reset:* Clear app data or reinstall the app between tests if necessary to return to a clean, default state.

- *Execution Steps:*

1. Navigate to the Medication Subcategory Section
    - Action: Open the app and navigate to the section displaying medication subcategory information.
    - Expected Result: The section loads without errors, presenting the medication sub-categories available for viewing.
2. Locate and Identify Medication Sub-Categories
    - Action: Identify and locate the different medication sub-categories displayed in the section.
    - Expected Result: Clear visibility of medication sub-categories with proper labeling and organization.
3. Verify Accuracy of Displayed Data
    - Action: Check each medication subcategory for the accuracy of the displayed data.
    - Expected Result: The app accurately presents medication information, matching the expected data set for each sub-category.
4. Assess Format and Readability
    - Action: Evaluate if the medication subcategory data is presented in a clear, understandable format and is easy to read.

- Expected Result: The format of the displayed data is consistent with design specifications, and readability standards are met.

5. Test User Interaction with Medication Sub-Categories
    - Action: Interact with the displayed medication sub-categories, such as tapping or selecting different sub-categories.
    - Expected Result: The app responds promptly to interactions, allowing users to access detailed information for each medication sub-category.

- *Execution Metrics:*
Placeholder first attempt to execute this test case date/time: [To be filled out during testing]
Placeholder number of attempts to successfully execute this test case: [To be filled out during testing]
Placeholder list of software defects discovered by this test case: [To be filled out during testing]
Placeholder is this test case executed successfully? (yes/no): [To be filled out during testing]

CID: 14 - Title: Nutrition Sub-Category Displayed

- *Description:* This use case aims to assess the app's capability to accurately display nutrition sub-category information. The focus is on ensuring that the app presents nutrition-related data correctly and adheres to the specified standards.

- *Development Phase:* This falls under the User Acceptance Testing (UAT) phase. UAT is crucial for confirming that the app meets end-users' requirements and expectations, specifically in displaying nutrition sub-category details.

- *Test Goals and Measurement:* The primary goal is to verify that the app accurately displays nutrition sub-category information. Measurement will be based on the accuracy of the displayed data, adherence to predefined format specifications, and the overall user experience in accessing nutrition sub-category details.

- Test Data/ Tools: A set of diverse data entries representing various nutrition sub-categories that the app is expected to display. Additionally, tools for tracking data accuracy and format compliance.
- Test Startup/Closedown/Reset Procedure:
    - *Startup:* Begin with the app launched and navigate to the section displaying nutrition sub-category information.
    - *Closedown:* Properly close the app and clear it from recent apps to ensure no data caching affects subsequent tests.
    - *Reset:* Clear app data or reinstall the app between tests if necessary to return to a clean, default state.

- Execution Steps:

1. Navigate to the Nutrition Sub-Category Section
    - Action: Open the app and navigate to the section displaying nutrition sub-category information.
    - Expected Result: The section loads without errors, presenting the nutrition sub-categories available for viewing.
2. Locate and Identify Nutrition Sub-Categories
    - Action: Identify and locate the different nutrition sub-categories displayed in the section.
    - Expected Result: Clear visibility of nutrition sub-categories with proper labeling and organization.
3. Verify Accuracy of Displayed Data
    - Action: Check each nutrition sub-category for the accuracy of the displayed data.
    - Expected Result: The app accurately presents nutrition information, matching the expected data set for each sub-category.
4. Assess Format and Readability
    - Action: Evaluate if the nutrition sub-category data is presented in a clear, understandable format and is easy to read.
    - Expected Result: The format of the displayed data is consistent with design specifications, and readability standards are met.

5. Test User Interaction with Nutrition Sub-Categories
    - Action: Interact with the displayed nutrition sub-categories, such as tapping or selecting different sub-categories.
    - Expected Result: The app responds promptly to interactions, allowing users to access detailed information for each nutrition sub-category.

- *Execution Metrics:*
Placeholder first attempt to execute this test case date/time: [To be filled out during testing]
Placeholder number of attempts to successfully execute this test case: [To be filled out during testing]
Placeholder list of software defects discovered by this test case: [To be filled out during testing]
Placeholder is this test case executed successfully? (yes/no): [To be filled out during testing]

CID: 15 - Title: Search-Bar Functionality

- *Description:* This use case aims to evaluate the functionality of the search bar within the app. The focus is on ensuring that the search bar operates as expected, allowing users to efficiently search for specific information within the app.

- *Development Phase:* This falls under the User Acceptance Testing (UAT) phase. UAT is essential for confirming that the app meets end-users' requirements and expectations, particularly regarding the search-bar functionality.

- *Test Goals and Measurement:* The primary goal is to verify that the search bar effectively allows users to search for information within the app. Measurement will be based on the accuracy and speed of search results, adherence to predefined search parameters, and overall user experience.

- Test Data/ Tools: Various data entries representing different types of information available within the app. Tools for tracking search accuracy and speed.
- Test Startup/Closedown/Reset Procedure:

- *Startup:* Begin with the app launched and ensure access to the home page or a relevant section for initiating searches.
- *Closedown:* Properly close the app and clear it from recent apps to ensure no data caching affects subsequent tests.
- *Reset:* Clear app data or reinstall the app between tests if necessary to return to a clean, default state.

- *Execution Steps:*

- Step: Access the Search Bar
  Action: Open the app and locate the search bar on the viewing page or in the designated section.
  Expected Result: The search bar is visible and ready for input.

- Step: Perform a Basic Search
  Action: Enter a known search term related to the data entries within the app and initiate the search.
  Expected Result: The app returns relevant search results quickly and accurately, matching the input term.

- Step: Test Search with Different Keywords
  Action: Repeat the search with various keywords, including common and uncommon terms, to test the range of the search function.
  Expected Result: The search function consistently provides relevant results for different keywords.

- Step: Check for Auto-suggest Functionality
  Action: Start typing a keyword in the search bar to check if auto-suggestions or auto-complete options appear.
  Expected Result: The app provides auto-suggestions or auto-completes the query, enhancing the search experience.

- Step: Validate Search Result Accuracy
  Action: Select a few search results to verify they correctly match the search query.
  Expected Result: Clicked search results are relevant to the query, indicating high accuracy of the search function.

- *Execution Metrics:*
Placeholder first attempt to execute this test case date/time: [To be filled out during testing]
Placeholder number of attempts to successfully execute this test case: [To be filled out during testing]
Placeholder list of software defects discovered by this test case: [To be filled out during testing]
Placeholder is this test case executed successfully? (yes/no): [To be filled out during testing]

CID: 16 - Title: Device Compatibility

- *Description:* This test case aims to evaluate the app's compatibility across multiple devices, ensuring a consistent and functional user experience on various platforms. The focus is on verifying that the app performs optimally and maintains its integrity when accessed from different devices.

- *Development Phase:* This falls under the User Acceptance Testing (UAT) phase. UAT is crucial for confirming that the app meets end-users' requirements and expectations across a diverse range of devices.

- *Test Goals and Measurement:* The primary goal is to ensure that the app is compatible with multiple devices. Measurement will be based on the successful execution of the app on different devices, adherence to device-specific design standards, and the absence of functionality issues.

- Test Data/ Tools: A variety of devices representing different operating systems, screen sizes, and resolutions. Additionally, tools for tracking any inconsistencies or issues related to device compatibility.
- Test Startup/Closedown/Reset Procedure:

- *Startup:* Begin by launching the app on a specified device, ensuring that it loads without errors and maintains a consistent user interface.
- *Closedown:* Properly close the app on each device, ensuring that there is no lingering data or caching affecting subsequent tests.
- *Reset:* If necessary, clear app data or reinstall the app between tests to return to a clean, default state on each device.

- Execution Steps:

1. Identify Compatible Devices
    - Action: Create a list of devices representing various operating systems, screen sizes, and resolutions for testing.
    - Expected Result: A diverse set of devices that mimic the potential user base.
2. Install and Launch the App on Each Device
    - Action: Install the app on each identified device and launch it.
    - Expected Result: The app installs successfully, and the user interface is consistent with design specifications on each device.
3. Navigate through App Features
    - Action: Test the app's navigation and functionality on each device, ensuring all features work as intended.
    - Expected Result: The app operates smoothly on each device, with no functionality issues or inconsistencies.
4. Test Device-Specific Features
    - Action: If the app has features specific to certain devices (e.g., gestures, sensors), test these features on the respective devices.
    - Expected Result: Device-specific features function correctly, enhancing the user experience on compatible devices.
5. Assess Display and Layout Consistency

- Action: Compare the app's display and layout across devices to ensure consistency in design and presentation.
- Expected Result: The app maintains a consistent and visually appealing layout on all tested devices.

- *Execution Metrics:*
Placeholder first attempt to execute this test case date/time: [To be filled out during testing]
Placeholder number of attempts to successfully execute this test case: [To be filled out during testing]
Placeholder list of software defects discovered by this test case: [To be filled out during testing]
Placeholder is this test case executed successfully? (yes/no): [To be filled out during testing]

CID: 17 - Title: Graph By Selected Category

- *Description:* This test case focuses on evaluating the app's capability to generate and display graphical representations based on a selected category. The goal is to ensure accurate data representation in the form of graphs, meeting predefined format specifications and providing a meaningful visual representation of the selected category.

- *Development Phase:* This falls under the User Acceptance Testing (UAT) phase. UAT is essential for confirming that the app meets end-users' requirements and expectations, specifically in the generation and display of graphs based on selected categories.

- *Test Goals and Measurement:* The primary goal is to verify that the app accurately generates and displays graphs based on the selected category. Measurement will be based on the accuracy of the displayed graph data, adherence to predefined format specifications, and the overall user experience in interpreting the visual representation.

- Test Data/ Tools: A set of diverse data entries representing various categories that the app is expected to generate graphs for. Additionally, tools for tracking graph accuracy and format compliance.

- Test Startup/Closedown/Reset Procedure:
    - *Startup:* Begin with the app launched and navigate to the section where graph generation based on a selected category is available.
    - *Closedown:* Properly close the app and clear it from recent apps to ensure no data caching affects subsequent tests.
    - *Reset:* Clear app data or reinstall the app between tests if necessary to return to a clean, default state.

- *Execution Steps:*

1. Navigate to the Graph Generation Section
    - Action: Open the app and navigate to the section where graph generation based on a selected category is available.
    - Expected Result: The section loads without errors, providing access to the functionality for generating graphs.
2. Identify and Select a Category
    - Action: Identify and select a category for which a graph needs to be generated.
    - Expected Result: Clear visibility of available categories, and the selected category is highlighted or indicated.
3. Generate Graph
    - Action: Initiate the process of generating a graph based on the selected category.
    - Expected Result: The app accurately processes the data and generates a graph representative of the selected category.
4. Verify Graph Accuracy
    - Action: Examine the generated graph to ensure it accurately represents the data for the selected category.
    - Expected Result: The graph accurately reflects the data associated with the selected category, meeting predefined format specifications.
5. Assess Graph Format and Readability
    - Action: Evaluate if the generated graph is presented in a clear, understandable format and is easy to read.

- Expected Result: The format of the graph is consistent with design specifications, and readability standards are met.

6. Test User Interaction with the Graph
    - Action: Interact with the generated graph, such as zooming, panning, or selecting specific data points.
    - Expected Result: The app responds promptly to interactions, providing a seamless and informative user experience with the graph.

- *Execution Metrics:*
Placeholder first attempt to execute this test case date/time: [To be filled out during testing]
Placeholder number of attempts to successfully execute this test case: [To be filled out during testing]
Placeholder list of software defects discovered by this test case: [To be filled out during testing]
Placeholder is this test case executed successfully? (yes/no): [To be filled out during testing]

CID: 18 - Title: Graph Display Model UI

- *Description:* This test case aims to evaluate the User Interface (UI) of the app's graph display model. The focus is on assessing the visual representation, layout, and overall user experience when viewing graphs. The goal is to ensure that the graph display model adheres to predefined design specifications, providing users with a clear and intuitive interface.

- *Development Phase:* This falls under the User Acceptance Testing (UAT) phase. UAT is essential for confirming that the app's graph display model meets end-users' requirements and expectations.

- *Test Goals and Measurement:* The primary goal is to verify that the app's graph display model UI is visually appealing, user-friendly, and aligns with design specifications. Measurement will be based on the adherence to UI design standards, clarity in graph presentation, and overall user satisfaction.

- Test Data/ Tools: A set of diverse data entries representing various graphs that the app is expected to display. Additionally, tools for tracking UI design adherence and user satisfaction.
- Test Startup/Closedown/Reset Procedure:
    - *Startup:* Begin with the app launched and navigate to the section where the graph display model is available.
    - *Closedown:* Properly close the app and clear it from recent apps to ensure no data caching affects subsequent tests.
    - *Reset:* Clear app data or reinstall the app between tests if necessary to return to a clean, default state.

- *Execution Steps:*

1. Navigate to the Graph Display Model Section
    - Action: Open the app and navigate to the section where the graph display model is available.
    - Expected Result: The section loads without errors, providing access to the graph display model UI.
2. Select a Graph to View
    - Action: Identify and select a graph to view within the app's graph display model.
    - Expected Result: Clear visibility of available graphs, and the selected graph is highlighted or indicated.
3. Assess Visual Representation
    - Action: Evaluate the visual representation of the selected graph, focusing on colors, labels, and overall clarity.
    - Expected Result: The graph is visually appealing, with clear differentiation of data points and adherence to design specifications.
4. Check Layout and Organization
    - Action: Examine the layout and organization of the graph display model UI, ensuring it is intuitive and user-friendly.
    - Expected Result: The UI layout is well-organized, providing users with an intuitive and easy-to-navigate experience.
5. Verify User Interaction with the Graph Display Model

- Action: Interact with the displayed graph, such as zooming, panning, or selecting specific data points.
- Expected Result: The app responds promptly to interactions, providing a seamless and engaging user experience with the graph display model.

6. Assess User Satisfaction
    - Action: Collect user feedback or conduct surveys to gauge overall satisfaction with the graph display model UI.
    - Expected Result: Positive user feedback, indicating satisfaction with the visual representation and usability of the graph display model.

- *Execution Metrics:*
Placeholder first attempt to execute this test case date/time: [To be filled out during testing]
Placeholder number of attempts to successfully execute this test case: [To be filled out during testing]
Placeholder list of software defects discovered by this test case: [To be filled out during testing]
Placeholder is this test case executed successfully? (yes/no): [To be filled out during testing]

CID: 19 - Title: Close Graph Functionality

- *Description:* This test case aims to evaluate the functionality of the "Close Graph" feature within the app's graph display model. The focus is on ensuring that users can effectively close and exit the displayed graph, providing a seamless and intuitive user experience.

- *Development Phase:* This falls under the User Acceptance Testing (UAT) phase. UAT is crucial for confirming that the app's "Close Graph" functionality meets end-users' expectations and contributes to an efficient user interface.

- *Test Goals and Measurement:* The primary goal is to verify that the "Close Graph" feature functions as expected, allowing users to exit the displayed graph without encountering errors. Measurement will be based on the successful execution of the "Close Graph" action and the absence of any associated defects.

- Test Data/ Tools: A set of diverse data entries representing various graphs that the app is expected to display. Additionally, tools for tracking the functionality of the "Close Graph" feature.
- Test Startup/Closedown/Reset Procedure:
    - *Startup:* Begin with the app launched and navigate to the section where the graph display model is available.
    - *Closedown:* Properly close the app and clear it from recent apps to ensure no data caching affects subsequent tests.
    - *Reset:* Clear app data or reinstall the app between tests if necessary to return to a clean, default state.

- *Execution Steps:*

1. Navigate to the Graph Display Model Section
    - Action: Open the app and navigate to the section where the graph display model is available.
    - Expected Result: The section loads without errors, providing access to the displayed graph.
2. View a Graph
    - Action: Select and view a graph within the app's graph display model.
    - Expected Result: The selected graph is displayed, and users can interact with it.
3. Initiate "Close Graph" Action
    - Action: Initiate the "Close Graph" action, typically by using a designated button or gesture.
    - Expected Result: The displayed graph closes without errors, returning users to the previous screen or the graph selection menu.
4. Verify Graph Closure
    - Action: Confirm that the graph is successfully closed and no longer visible on the screen.
    - Expected Result: The graph is no longer displayed, and the app interface reflects the closure without glitches.

5. Test "Close Graph" with Multiple Open Graphs
    - Action: If the app supports multiple open graphs simultaneously, test the "Close Graph" action with multiple graphs.
    - Expected Result: The "Close Graph" feature handles multiple open graphs gracefully, closing the selected graph without affecting others.
6. Assess User Experience
    - Action: Evaluate the overall user experience of using the "Close Graph" feature, considering ease of use and responsiveness.
    - Expected Result: Users can easily close graphs, and the action is responsive, contributing to a positive user experience.

- *Execution Metrics:*
Placeholder first attempt to execute this test case date/time: [To be filled out during testing]
Placeholder number of attempts to successfully execute this test case: [To be filled out during testing]
Placeholder list of software defects discovered by this test case: [To be filled out during testing]
Placeholder is this test case executed successfully? (yes/no): [To be filled out during testing]

CID: 20 - Title: Graph Display Time and Quantity of Input

- *Description:* This test case focuses on evaluating the accuracy and responsiveness of the app's graph display concerning time-based data and the quantity of input. The goal is to ensure that the graph accurately represents time-related data and handles varying quantities of input without compromising performance.

- *Development Phase:* This falls under the User Acceptance Testing (UAT) phase. UAT is essential for confirming that the app's graph display model effectively handles time-based data and various quantities of input.

- *Test Goals and Measurement:* The primary goal is to verify that the app's graph display accurately represents time-related data and can handle varying quantities of input without performance issues. Measurement will be based on

the accuracy of time-related data representation, the responsiveness of the graph to input changes, and the overall user experience.

- Test Data/ Tools: A set of diverse data entries representing time-based information with varying quantities of input. Additionally, tools for tracking the accuracy of time-related data and the performance of the graph display.
- Test Startup/Closedown/Reset Procedure:
    - *Startup:* Begin with the app launched and navigate to the section where the graph display is available for time-based data with varying input quantities.
    - *Closedown:* Properly close the app and clear it from recent apps to ensure no data caching affects subsequent tests.
    - *Reset:* Clear app data or reinstall the app between tests if necessary to return to a clean, default state.

- *Execution Steps:*

1. Navigate to the Graph Display Section for Time-Based Data
    - Action: Open the app and navigate to the section where the graph display for time-based data is available.
    - Expected Result: The section loads without errors, providing access to the graph display.
2. Select a Graph with Time-Based Data
    - Action: Choose a graph that represents time-based data within the app's graph display model.
    - Expected Result: The selected graph accurately represents time-related information, and users can interact with it.
3. Input Varying Quantities of Data
    - Action: Input different quantities of data into the graph, simulating varying input scenarios.
    - Expected Result: The graph accurately reflects the input data, and the display adjusts dynamically to varying quantities without errors.
4. Verify Time-Related Data Accuracy

- Action: Check the accuracy of time-related data representation on the graph.
- Expected Result: Time-related data is accurately displayed, and the graph aligns with expected time intervals.

5. Assess Responsiveness to Input Changes
    - Action: Continuously input data changes to the graph and evaluate the responsiveness of the display.
    - Expected Result: The graph responds promptly to input changes, providing real-time updates without lag or delay.
6. Test Graph Performance with High Quantity of Input
    - Action: Input a high quantity of data into the graph to test performance under increased load.
    - Expected Result: The app handles a high quantity of input without significant performance degradation, maintaining a responsive and smooth user experience.

- *Execution Metrics:*
Placeholder first attempt to execute this test case date/time: [To be filled out during testing]
Placeholder number of attempts to successfully execute this test case: [To be filled out during testing]
Placeholder list of software defects discovered by this test case: [To be filled out during testing]
Placeholder is this test case executed successfully? (yes/no): [To be filled out during testing]

Add Page

CID: 21 - Title: Add Page Navigation Button

- *Description:* This test case aims to verify the functionality and reliability of the navigation button designed to take users to the page where they can add new data into the app. It focuses on the button's responsiveness and navigational accuracy

- *Development Phase:* This test is placed in the System Testing phase as it

involves testing the integrated functionality of the button within the app, ensuring it interacts correctly with other components and leads the user to the correct page.

- *Test Goals and Measurement:* Ensure the "Add Page" navigation button quickly and accurately takes the user to the appropriate data entry page without errors. Success is determined by the button's responsiveness (action executed within 1-2 seconds).

- Test Data/ Tools: No specific data is required for this test case. Test app functionality with an Android emulator tool.
- Test Startup/Closedown/Reset Procedure:
    - *Startup:* Begin with the app launched and logged in.
    - *Closedown:* Properly exit the app and clear it from the recent apps list to ensure no residual data affects the next test.
    - *Reset:* Clear the app's data or reinstall the app to ensure a fresh state for each test if necessary.

- *Execution Steps:*
    - Step: Locate and interact with the "Add Page" navigation button.
      Action: From the starting page, click or tap on the "Add Page" navigation button.
      Expected Result: The app responds immediately (within 1-2 seconds) and navigates to the data entry page.

    - Step: Verify the correct page navigation.
      Action: Observe the screen to confirm you are taken to the correct page for adding new data.
      Expected Result: The data entry page loads successfully, indicating the correct page navigation.

    - Step: Check readiness for data entry.

Action: Review the data entry page for any input fields, buttons, or instructions that indicate the page is ready for the user to begin entering data.
Expected Result: Input fields and relevant UI elements are visible and ready for interaction, confirming the page's readiness for data entry.

- *Execution Metrics:*
Placeholder first attempt to execute this test case date/time: [To be filled out during testing]
Placeholder number of attempts to successfully execute this test case: [To be filled out during testing]
Placeholder list of software defects discovered by this test case: [To be filled out during testing]
Placeholder is this test case executed successfully? (yes/no): [To be filled out during testing]

CID: 22 - Title: Edit Health Data Entry

- *Description:* This test case focuses on verifying the app's ability to allow users to edit previously entered health data. It assesses the functionality for loading the correct data for editing, the responsiveness of the editing interface, the accuracy of data saving, and the immediate reflection of edited data within the app.

- *Development Phase:* This test case falls under the Integration Testing phase. It involves interaction between various components of the app, including the UI, and database.

- *Test Goals and Measurement:* Ensure users can successfully edit their health data entries with changes accurately saved and reflected within the app. Success is measured by the edit function's responsiveness, the accuracy of the data load for editing, the correct saving of edits, and the immediate update of these edits in the app's data display.

- Test Data/ Tools: Pre-existing health data entries in the app. Create a variety of health data entries to cover different types and formats of data that users might edit. Test app functionality with an Android emulator tool.

- Test Startup/Closedown/Reset Procedure:
    - *Startup:* Launch the app and navigate to the “Add Page” where health data entries are displayed. Select an entry to edit.
    - *Closedown:* Close the app and clear it from the recent apps list to ensure a fresh environment for the next test run.
    - *Reset:* If necessary, clear the app's data or reinstall the app to reset all settings and data to their default states.

- Execution Steps:

- Step: Select a health data entry to edit.
  Action: From the list of displayed health data entries, choose one and initiate the edit action.
  Expected Result: The selected data entry loads into an editing interface, displaying the current data accurately for modification.

- Step: Modify the data in the entry.
  Action: Change several fields of the selected health data entry, such as values or descriptions.
  Expected Result: Modifications are allowed, and all changes are accurately reflected in the editing interface.

- Step: Save the edited data.
  Action: After making the desired changes, save the edited data entry.
  Expected Result: The app responds promptly to the save action, and no errors occur during the saving process.

- Step: Verify that updated data is displayed.
  Action: Return to the view where the health data entries are listed or displayed.
  Expected Result: The edited data entry is immediately updated in the list, reflecting all changes made.

- Execution Metrics:

Placeholder first attempt to execute this test case date/time: [To be filled out during testing]

Placeholder number of attempts to successfully execute this test case: [To be filled out during testing]
Placeholder list of software defects discovered by this test case: [To be filled out during testing]
Placeholder is this test case executed successfully? (yes/no): [To be filled out during testing]

CID: 23 - Title: Add New Data Button Functionality

- *Description:* This test case aims to validate the functionality of the "Add New Data" button within the app, ensuring it properly initiates the process for users to input new data entries. The focus will be on the button's responsiveness, the correct navigation to the data entry interface, and the readiness of the interface for data input.

- *Development Phase:* This test is categorized under the System Testing phase, as it involves verifying the integrated functionality of the button within the app's ecosystem, ensuring it leads the user to the correct interface for adding new data.

- *Test Goals and Measurement:* Confirm that the buttons are functional, guiding the user to the data entry interface promptly and accurately, with the interface being ready for immediate data input. The button's success is measured by its responsiveness (action executed within 1-2 seconds), navigational accuracy (leading to the data entry interface 100% of the time), and the data entry interface's readiness for input upon navigation.

- Test Data/ Tools: Test Data is not specifically required for this test case. Test app functionality with an Android emulator tool.
- Test Startup/Closedown/Reset Procedure:
    - *Startup:* Begin with the app launched, ensuring you are at the “Add” page
    - *Closedown:* Exit the app and remove it from the recent applications list to ensure no residual data affects subsequent tests.
    - *Reset:* If necessary, clear the app's data or reinstall the app to ensure a fresh state for each test iteration.

- *Execution Steps:*
    - Step: Locate and interact with the buttons.
      Action: Identify the "Add New Data" buttons on the current screen and initiate the add new data process by clicking or tapping on the button.
      Expected Result: The app responds immediately (within 1-2 seconds) and navigates to the data entry interface.

    - Step: Verify correct navigation to the data entry interface.
      Action: Confirm that you are taken to the correct interface designed for adding new data entries.
      Expected Result: The data entry interface loads successfully, indicating correct navigation.

    - Step: Assess readiness for data entry.
      Action: Evaluate the data entry interface for the presence of input fields, instructions, or any UI elements indicating the page is ready for the user to begin inputting data.
      Expected Result: All necessary UI elements for data entry are visible and ready for interaction, confirming the interface's readiness for new data input.
    -

- *Execution Metrics:*
Placeholder first attempt to execute this test case date/time: [To be filled out during testing]
Placeholder number of attempts to successfully execute this test case: [To be filled out during testing]
Placeholder list of software defects discovered by this test case: [To be filled out during testing]
Placeholder is this test case executed successfully? (yes/no): [To be filled out during testing]

CID: 24 - Title: Remove Data Button Functionality

- *Description:* This test case is designed to verify the functionality of the "Remove Data" button, ensuring that it properly deletes specific data entries from the app. The focus will be on the button's responsiveness, the accuracy of the

deletion process, and the app's ability to update the user interface to reflect the removal of data.

- *Development Phase:* This test falls under the Integration Testing phase, as removing data typically involves interactions between the user interface, application logic, and backend databases

- *Test Goals and Measurement:* Ensure the "Remove Data" button effectively deletes the selected data entries and that the user interface is promptly updated to reflect these changes. Success is measured by the button's responsiveness, the correct and complete removal of targeted data entries, and the UI's immediate update to indicate the removal.

- Test Data/ Tools: Pre-populate the application with dummy data entries that can be safely removed during testing. Test app functionality with an Android emulator tool.
- Test Startup/Closedown/Reset Procedure:
    - *Startup:* Start with the application launched and navigate to the Add page where data entries are displayed and can be removed.
    - *Closedown:* Properly exit the application and clear it from recent applications to ensure a fresh start for subsequent tests.
    - *Reset:* Clear the app's data.

- *Execution Steps:*

- Step: Identify and select a data entry for removal.
  Action: Navigate to the list of data entries, choose one that is designated for testing, and initiate the removal process by clicking or tapping the "Remove Data" button.
  Expected Result: A prompt or confirmation message appears, asking to confirm the deletion of the selected data entry.

- Step: Confirm the removal of the data entry.
  Action: Confirm the deletion operation through the app's user interface.

Expected Result: The app responds promptly to the confirmation action, and the data entry is removed from the list.

- Step: Verify the data entry has been successfully removed.
  Action: Review the list of data entries to ensure the removed entry no longer appears.
  Expected Result: The user interface updates to reflect the removal, and the specific data entry is not visible in the list.

- *Execution Metrics:*
Placeholder first attempt to execute this test case date/time: [To be filled out during testing]
Placeholder number of attempts to successfully execute this test case: [To be filled out during testing]
Placeholder list of software defects discovered by this test case: [To be filled out during testing]
Placeholder is this test case executed successfully? (yes/no): [To be filled out during testing]

CID: 25 - Title: Preview Print Button Functionality

- *Description:* Test the functionality of the Preview Print button to ensure it correctly generates a print preview of the selected data or document within the app.

- *Development Phase:* User Interface Testing phase, focusing on the button's ability to interact with the print service and generate a preview as expected.

- *Test Goals and Measurement:* Confirm that the Preview Print button triggers a print preview screen, displaying the document or data accurately as it would appear in a printout. Success is determined by the immediate and accurate display of a print preview upon button activation.

- Test Data/ Tools: Sample data or documents available within the app for printing. Test app functionality with an Android emulator tool.
- Test Startup/Closedown/Reset Procedure:

- *Startup:* Open the app to the section with printable content and locate the Preview Print button.
- *Closedown:* Close the app and clear it from recent applications.
- *Reset:* Not required.

- *Execution Steps:*

- Step: Navigate to a document or data set eligible for printing within the app.

- Step: Click the Preview Print button.

- Step: Verify a print preview screen appears, accurately displaying the content intended for print.

- *Execution Metrics:*
Placeholder first attempt to execute this test case date/time: [To be filled out during testing]
Placeholder number of attempts to successfully execute this test case: [To be filled out during testing]
Placeholder list of software defects discovered by this test case: [To be filled out during testing]
Placeholder is this test case executed successfully? (yes/no): [To be filled out during testing]

CID: 26 - Title: Add New Data Sub-Category Navigation

- *Description:* This test case aims to verify the app's functionality that allows users to navigate through sub-categories when adding new data. It assesses the ease of navigation, accuracy of sub-category listings, and the system's ability to display the correct input fields for each sub-category.

- *Development Phase:* This test is placed in the Integration Testing phase, as it involves testing the interaction between the app's navigation system and the data entry interface, ensuring seamless operation across different parts of the app.

- *Test Goals and Measurement:* Ensure that users can smoothly navigate through sub-categories for data entry, with each sub-category accurately displaying the appropriate input fields. Success is measured by the navigation's intuitiveness, accuracy of sub-category listings, and the correct display of input fields relevant to each sub-category.

- Test Data/ Tools: A list of sub-categories that should be available for selection when adding new data, along with the expected input fields for each sub-category. Test app functionality with an Android emulator tool.
- Test Startup/Closedown/Reset Procedure:
    - *Startup:* Begin with the app launched and navigate to the page where the "Add New Data" functionality is accessible.
    - *Closedown:* Close the app and remove it from recent applications to ensure a fresh start for each test.
    - *Reset:* Clear the app's data or reinstall the app if necessary to ensure a clean state for each test iteration

- *Execution Steps:*

- Step: Initiate the "Add New Data" process.
  Action: Click or tap on the "Add New Data" buttons within the app.
  Expected Result: The app presents a list of sub-categories for data entry.

- Step: Navigate through sub-categories.
  Action: Select a sub-category from the list provided.
  Expected Result: The app navigates to the correct sub-category, displaying the appropriate input fields for data entry.

- Step: Verify input fields for a sub-category.
  Action: Review the displayed input fields to ensure they match the expected fields for the selected sub-category.
  Expected Result: All expected input fields for the selected sub-category are displayed correctly, allowing for accurate data entry.

- Step: Test navigation back to the sub-category list.
  Action: Use the app's navigation to return to the list of subcategories after selecting one.
  Expected Result: The app allows users to return to the sub-category list easily, facilitating navigation between different sub-categories without losing context.
-

- *Execution Metrics:*
Placeholder first attempt to execute this test case date/time: [To be filled out during testing]
Placeholder number of attempts to successfully execute this test case: [To be filled out during testing]
Placeholder list of software defects discovered by this test case: [To be filled out during testing]
Placeholder is this test case executed successfully? (yes/no): [To be filled out during testing]

CID: 27 - Title: Data Entry Model and Exit Functionality

- *Description:* This test case assesses the functionality of the data entry modal, including its ability to open correctly, accept data input, and close or exit without data loss or system errors.

- *Development Phase:* User Interface Testing phase, focusing on the model's interaction with users, including input fields, submission actions, and the ability to exit the modal safely.

- *Test Goals and Measurement:* Ensure the data entry modal opens, allows for data input, and exits correctly, preserving entered data as expected or discarding it appropriately based on user actions. Modal responsiveness, data integrity upon exit, and error-free closure.

- Test Data/ Tools: Sample data inputs for all fields within the modal. Test app functionality with an Android emulator tool.
- Test Startup/Closedown/Reset Procedure:

- *Startup:* Launch the app and navigate to the feature that triggers the data entry modal.
- *Closedown:* Close the app after testing and clear from recent apps.
- *Reset:* Clear app data or reinstall for a fresh state if needed.

- *Execution Steps:*

- Step: Open Modal Trigger the modal to open and verify it appears as expected.
- Step: Input Data Enter sample data into all fields and attempt to submit/save.
- Step Exit Modal Test the exit functionality, both with and without saving data, to ensure the modal closes correctly and data is handled as intended.
- Step Verify Data If applicable, confirm that saved data appears correctly in the app after modal closure.

- *Execution Metrics:*
Placeholder first attempt to execute this test case date/time: [To be filled out during testing]
Placeholder number of attempts to successfully execute this test case: [To be filled out during testing]
Placeholder list of software defects discovered by this test case: [To be filled out during testing]
Placeholder is this test case executed successfully? (yes/no): [To be filled out during testing]

CID: 28 - Title: Button Click UI

- *Description:* Evaluates the visual and functional response of buttons within the app upon user interaction, ensuring buttons are responsive and provide visual feedback.

- *Development Phase:* User Interface Testing phase, with a focus on the responsiveness and feedback of UI elements to user actions.

- *Test Goals and Measurement:* Confirm that all buttons in the app respond visually to clicks/taps and trigger their associated actions without delay. Visual feedback on interaction (e.g., color change, animation) and action execution time.

- Test Data/ Tools: Test Data is not applicable. Test app functionality with an Android emulator tool.
- Test Startup/Closedown/Reset Procedure:
    - *Startup:* Open the app to a screen with actionable buttons.
    - *Closedown:* Close the app and clear it from the device's memory.
    - *Reset:* Reinstall or clear data as needed for a clean test environment.
    -

- *Execution Steps:*

- Step: Identify Buttons Locate buttons to be tested on the current screen.
  Action: Review the current screen or interface within the app to locate all buttons that are intended for user interaction.
  Expected Result: All buttons on the screen should be visible and clearly identifiable as interactive elements.

- Step: Click Buttons to interact with each button, noting the visual feedback and time to action.
  Action: Individually interact with each identified button by clicking or tapping on them.
  Expected Result: Upon interaction, each button should provide immediate visual feedback to indicate it has been pressed.

- Step: Verify Actions to ensure the intended action for each button is executed correctly.
  Action: After interacting with each button, verify that the intended action for that button is executed correctly.

Expected Outcome: The app correctly executes the intended action for each button without errors or delays.

- *Execution Metrics:*
Placeholder first attempt to execute this test case date/time: [To be filled out during testing]
Placeholder number of attempts to successfully execute this test case: [To be filled out during testing]
Placeholder list of software defects discovered by this test case: [To be filled out during testing]
Placeholder is this test case executed successfully? (yes/no): [To be filled out during testing]

CID: 29 - Title: Data Input

- *Description:* This test case assesses the functionality and user experience of data input fields within the app, focusing on the ease of use, validation, and error handling.

- *Development Phase:* This test is part of the User Interface Testing phase, aiming to ensure that data input by users is efficiently captured, validated, and provides feedback when necessary.

- *Test Goals and Measurement:* Verify that all data input fields accept valid input and correctly reject invalid input with appropriate error messages. Success is measured by the accuracy of validation mechanisms and the clarity of user feedback.

- Test Data/ Tools: Test data should include a variety of valid and invalid inputs for each field. Use the app on a physical device or emulator for testing.
- Test Startup/Closedown/Reset Procedure:
    - *Startup:* Launch the app and navigate to the feature requiring data input.
    - *Closedown:* Close the app after completing the test.
    - *Reset:* If necessary, clear the input fields or restart the test environment to its default state before starting the test.

- *Execution Steps:*

  - Step: Navigate to the data input section.
    Action: Launch the app and find the section where data input is required.
    Expected Result: The data input section is accessible and ready for input.

  - Step: Input valid data.
    Action: Enter valid data into the input fields.
    Expected Result: The app accepts the data without errors.

  - Step: Input invalid data.
    Action: Enter invalid data into the input fields.
    Expected Result: The app rejects the data and displays an appropriate error message.

- *Execution Metrics:*
Placeholder first attempt to execute this test case date/time: [To be filled out during testing]
Placeholder number of attempts to successfully execute this test case: [To be filled out during testing]
Placeholder list of software defects discovered by this test case: [To be filled out during testing]
Placeholder is this test case executed successfully? (yes/no): [To be filled out during testing]

CID: 30 - Title: Data Save Functionality

- *Description:* This test case evaluates the app's ability to correctly save data entered by the user, ensuring data persistence across sessions.

- *Development Phase:* This test is incorporated into the Integration Testing phase, focusing on the interaction between the app's user interface and its underlying data storage mechanisms.

- *Test Goals and Measurement:* Confirm that data entered into the app is

accurately saved and persists across app restarts or device reboots. Success is measured by the integrity and persistence of saved data

- Test Data/ Tools: No specific test data required. Utilize the app on a suitable device or emulator for testing purposes.
- Test Startup/Closedown/Reset Procedure:
    - *Startup:* Open the app and navigate to the feature that allows data saving.
    - *Closedown:* Close the app after testing and then reopen it to check data persistence.
    - *Reset:* Clear the app's data or uninstall and reinstall the app if necessary to ensure a clean testing environment for each test run.

- Execution Steps:

- Step: Enter and save data.
  Action: Input data into the app and use the feature to save it.
  Expected Result: The app indicates that data saving was successful.

- Step: Verify data persistence.
  Action: Close and reopen the app, then navigate to the section where the data was saved.
  Expected Result: Previously saved data is accurately retrieved and displayed.

- Execution Metrics:
Placeholder first attempt to execute this test case date/time: [To be filled out during testing]
Placeholder number of attempts to successfully execute this test case: [To be filled out during testing]
Placeholder list of software defects discovered by this test case: [To be filled out during testing]
Placeholder is this test case executed successfully? (yes/no): [To be filled out during testing]

Profile Page

CID: 31 - Title: Profile Page Navigation Button

- *Description:* Validate the functionality and responsiveness of the navigation button on the profile page, ensuring the users can access and navigate through profile-related features accurately and quickly.

- *Development Phase:* Integration Testing. This phase is chosen because it involves testing the integration of individual components to create functionality across all individual navigation buttons.

- *Test Goals and Measurement:* Confirm that the navigation button accurately directs users to various profile components (Authentication, Profile Image, Name, Phone, Email) with no errors. Measured by performance and accuracy on each navigation.

- Test Data/ Tools: Manual Testing, Automated UI Testing frameworks
    - *Startup:* Ensure the user is logged into the app before starting tests. Navigate to the relevant feature for the test.
    - *Closedown:* Close the app after completing the tests.
    - *Reset:* Reset the testing environment by navigating back to the main page after each test, ensuring a clean state for subsequent tests.

- *Execution Steps:*

1. Access the Profile Page
2. Utilize the navigation button to explore each available section in the app.
3. Confirm that each section is accessible and the navigation functions as intended, maintaining accuracy and performance.

- *Execution Metrics:* Test is successful if all profile components are accessible using navigation with correct functions and display.

Placeholder first attempt to execute this test case date/time: [To be filled out during testing]

Placeholder number of attempts to successfully execute this test case: [To be filled out during testing]
Placeholder list of software defects discovered by this test case: [To be filled out during testing]
Placeholder is this test case executed successfully? (yes/no): [To be filled out during testing]

CID: 32 - Title: Sub-Account Permissions

- *Description:* Test the applications handling of sub-account permissions, verifying that each sub-account has the correct access levels defined by the main user.

- *Development Phase:* System Testing as this stage is crucial for verifying the applications overall functionality and its compliance with user requirements, while still ensuring security and reliability.

- *Test Goals and Measurement:* Ensure that sub-accounts are correctly limited or granted access based on the main/parent account permission settings. Measurement by the successful execution of permission-restricted actions.

- Test Data/ Tools: Test accounts with varied permission levels, backend access for permission verification.
- Test Startup/Closedown/Reset Procedure:
    - *Startup:* Set up sub-accounts with predefined permissions specific to the testing scope.
    - *Closedown:* Close the testing session and ensure all data is saved as required.
    - *Reset:* Restore permissions for all sub-accounts to default settings after testing to ensure a clean environment for future tests.

- *Execution Steps:*

1. Log in with sub-accounts
2. Attempt to access features based on the set granted and restricted permissions

3. Verify that access matches and functions correctly with the permission settings defined.

- *Execution Metrics:* Test is successful if sub-accounts can only access features aligned with their permissions and are unable to deviate or work-around permission restrictions.
Placeholder first attempt to execute this test case date/time: [To be filled out during testing]
Placeholder number of attempts to successfully execute this test case: [To be filled out during testing]
Placeholder list of software defects discovered by this test case: [To be filled out during testing]
Placeholder is this test case executed successfully? (yes/no): [To be filled out during testing]

CID: 33 - Title: Profile Picture

- *Description:* Ensure that users can upload, change, and delete their profile picture quickly and accurately, with the changes reflected immediately across the app.

- *Development Phase:* Functional Testing to assess specific functionalities of the application, like user interactions with uploading a profile picture.

- *Test Goals and Measurement:* Profile picture updates should be user-friendly with different image types accepted. These changes should take place instantly and application wide. Measurement by successful upload, update, and deletion.

- Test Data/ Tools: Sample images of different sizes and formats, automated image upload testing tools.
- Test Startup/Closedown/Reset Procedure:
    - *Startup:* Ensure a user profile exists for testing, with the ability to upload a profile picture.
    - *Closedown:* Log out of the application and clear temporary files.
    - *Reset:* Remove any test images uploaded and revert back to the original/default profile picture.

- *Execution Steps:*

1. Upload a new profile picture
2. Change the profile picture to a different image
3. Delete the profile picture after analyzing

- *Execution Metrics:* Success is marked by the app's ability to handle profile picture uploads, changes and deletions without issues, and that all supported formats and sizes are successfully accepted.
Placeholder first attempt to execute this test case date/time: [To be filled out during testing]
Placeholder number of attempts to successfully execute this test case: [To be filled out during testing]
Placeholder list of software defects discovered by this test case: [To be filled out during testing]
Placeholder is this test case executed successfully? (yes/no): [To be filled out during testing]

CID: 34 - Title: Edit Profile Functionality

- *Description:* Verify the edit profile functionality allows users to update their profile information accurately and that the changes are saved correctly and updated instantly in the server.

- *Development Phase:* User Acceptance Testing. This phase was chosen as it is crucial for validating the application against user requirements and expectations when managing profiles.

- *Test Goals and Measurement:* User profile updates should be quick and straightforward, to ensure accessibility, with all changes saved and reflected in the app. Measured by successful information update and retrieval, and the correctness of saved information without any errors displayed.

- Test Data/ Tools: Pre-defined data for profile updates, database verification tools.
- Test Startup/Closedown/Reset Procedure:
    - *Startup:* User profiles are pre-populated with editable information.

- *Closedown:* Clear any changes made during testing, log out of the application.
- *Reset:* Restore original profile information to ensure a consistent starting point for future tests.

- *Execution Steps:*

1. Access the Edit Profile section in the Profile Page
2. Update Name, Phone, Email information
3. Save changes and verify updates are correctly reflected in the application

- *Execution Metrics:* Test is considered successful if all profile updates are accurately saved to the database and displayed in the application instantly and without errors.
Placeholder first attempt to execute this test case date/time: [To be filled out during testing]
Placeholder number of attempts to successfully execute this test case: [To be filled out during testing]
Placeholder list of software defects discovered by this test case: [To be filled out during testing]
Placeholder is this test case executed successfully? (yes/no): [To be filled out during testing]

CID: 35 - Title: Profile Settings

- *Description:* Ensure the functionality and user accessibility of profile settings, including customization options and user preference settings.

- *Development Phase:* Usability Testing as this phase assesses the application's ease of use, with how many different settings there are and interchangeable with the average user.

- *Test Goals and Measurement:* Profile settings should be easily navigable, with all of the customization and preference settings on the user account functioning as expected. Measurement by successful application of setting changes, focussing on performance and reliability.

- Test Data/ Tools: Manual testing checklist through each available option, user/stakeholder survey tools.
- Test Startup/Closedown/Reset Procedure:
    - *Startup:* User logged into their profile with default settings applied.
    - *Closedown:* Log out and clear any remaining session data to ensure clean testing for the next tests.
    - *Reset:* Revert any changes made to the settings back to default states.

- *Execution Steps:*

1. Navigate to Profile Settings.
2. Explore and adjust various settings and customization options.
3. Verify changes are applied and persist within the app.

- *Execution Metrics:* Successful if all settings changes are accurately reflected and saved in the server to enhance the user experience per their preferences.
Placeholder first attempt to execute this test case date/time: [To be filled out during testing]
Placeholder number of attempts to successfully execute this test case: [To be filled out during testing]
Placeholder list of software defects discovered by this test case: [To be filled out during testing]
Placeholder is this test case executed successfully? (yes/no): [To be filled out during testing]

CID: 36 - Title: Share Functionality

- *Description:* Test the share functionality to ensure that users can easily share content from the app through various platforms and options without issues or data loss.

- *Development Phase:* Integration Testing as this phase is selected to test the integration of the application's internal sharing feature that should be seamless and reliable.

- *Test Goals and Measurement:* Sharing through the app should be designed to

be seamless across different platforms, with content correctly formatted depending on the user's selection. Measurement by successful shares and correct content format.

- Test Data/ Tools: Cross-platform sharing test cases, manual content format verification
- Test Startup/Closedown/Reset Procedure:
    - *Startup:* Ensure the application contains content that is available for sharing.
    - *Closedown:* Clear any shared content logs and sign out.
    - *Reset:* Return application to a state before test shares were initiated.

- *Execution Steps:*

1. Select content within the app to share or format to be downloaded
2. Share/download content through different options and platforms
3. Verify that the content shared and downloaded is correct and maintains the users' desired format.

- *Execution Metrics:* Test is successful if content is accurately shared across platforms, and downloaded correctly according to the user preference/selection, maintaining the correct information.
Placeholder first attempt to execute this test case date/time: [To be filled out during testing]
Placeholder number of attempts to successfully execute this test case: [To be filled out during testing]
Placeholder list of software defects discovered by this test case: [To be filled out during testing]
Placeholder is this test case executed successfully? (yes/no): [To be filled out during testing]

CID: 37 - Title: Support Us Functionality

- *Description:* Ensure the "Support Us" feature functions correctly, allowing users to contribute through the app easily and securely.

- *Development Phase:* Security Testing as this phase will ensure that the contribution process is secure, having sensitive user data and their financial information at risk.

- *Test Goals and Measurement:* Contributions using the "Support Us" feature should be straightforward and secure, and give user's clear instructions and feedback when making a contribution. Measurement by successful and secure transactions and user feedback.

- Test Data/ Tools: Payment gateway testing, usability testing checklist.
- Test Startup/Closedown/Reset Procedure:
    - *Startup:* User profile set up for financial transactions.
    - *Closedown:* Log out of the application and ensure no sensitive data remains.
    - *Reset:* Refund any mock contributions and reset financial data back to original.

- *Execution Steps:*

1. Navigate to "Support Us" in the application
2. Follow the instruction to contribute to our team
3. Verify that the contribution process is user-friendly and maintains maximum security.

- *Execution Metrics:* Success is measured by how quick and easy users can complete a contribution, and by reviewing user feedback on the process. There should also be no security flaws in every test.
Placeholder first attempt to execute this test case date/time: [To be filled out during testing]
Placeholder number of attempts to successfully execute this test case: [To be filled out during testing]
Placeholder list of software defects discovered by this test case: [To be filled out during testing]
Placeholder is this test case executed successfully? (yes/no): [To be filled out during testing]

CID: 38 - Title: Sub-Profile Navigation

- *Description:* Test the navigation within sub-profiles to ensure users can easily switch between, and manage multiple profiles and sub-profiles within the application.

- *Development Phase:* Usability Testing, as this phase is chosen to evaluate the ease of use and user satisfaction when navigating and managing sub-profiles.

- *Test Goals and Measurement:* Sub-profile navigation should be intuitive and quick, with users able to switch between profiles without confusion, error, or loss of progress or data after switching. Measurement by successful profile switches and user navigation experience.

- Test Data/ Tools: Manual testing procedures, user/stakeholder experience surveys
- Test Startup/Closedown/Reset Procedure:
    - *Startup:* User logged in with multiple sub-profiles created for testing.
    - *Closedown:* Log out and clear any session or user data.
    - *Reset:* Remove test sub-profiles and revert back to the original profile.

- *Execution Steps:*

1. Access the main user profile
2. Navigate to and switch between sub-profile and mark what happens to data and progress during a switch
3. Manage (add, edit, delete) sub-profiles.

- *Execution Metrics:* Test is successful if navigation and management of sub-profiles are executed without issues, enhancing user experience.
Placeholder first attempt to execute this test case date/time: [To be filled out during testing]

Placeholder number of attempts to successfully execute this test case: [To be filled out during testing]
Placeholder list of software defects discovered by this test case: [To be filled out during testing]
Placeholder is this test case executed successfully? (yes/no): [To be filled out during testing]

Settings Page

CID: 39 - Settings Page Navigation Button

- *Description:* Verify that the navigation button on the Settings page functions correctly, allowing users to access and modify different pages.

- *Development Phase:* Integration Testing as this phase ensures that the Settings page components integrate well and are functioning as expected.

- *Test Goals and Measurement:* Our goal is to ensure the navigation button leads to the correct settings options without errors or major delays. Measure each successful display and look for discrepancies in system response time, and user feedback.

- Test Data/ Tools: Manual testing, UI Testing tools
- Test Startup/Closedown/Reset Procedure:
    - *Startup:* User logged into the application, with the settings page accessible.
    - *Closedown:* Log out and clear cache/data to ensure a clean state for further testing.
    - *Reset:* Navigate back to the main Settings page after testing.

- *Execution Steps:*

1. App open on home screen
2. Navigate to Settings page

3. Click on each option (Logout, Customize User, Switch User) and verify each function executes correctly.
4. Verify back navigation works for each page.

- *Execution Metrics:* Test is successful if all options are accessible and functional using navigation, and see no errors during the process.
Placeholder first attempt to execute this test case date/time: [To be filled out during testing]
Placeholder number of attempts to successfully execute this test case: [To be filled out during testing]
Placeholder list of software defects discovered by this test case: [To be filled out during testing]
Placeholder is this test case executed successfully? (yes/no): [To be filled out during testing]

CID: 40 - Title: Data Export Functionality

- *Description:* Assess the app's ability to export user data correctly and securely upon request.

- *Development Phase:* System Testing. This phase was chosen to ensure that the data export functionality works correctly across the system and meets all security standards in place.

- *Test Goals and Measurement:* To confirm data export executes correctly, accurately saves information and maintains privacy. Measurement would be the successful export and data protection of all user information.

- Test Data/ Tools: Sample user data, Encryption and security testing tools
- Test Startup/Closedown/Reset Procedure:
    - *Startup:* User profile filled with data for export.
    - *Closedown:* Securely delete exported data and log out of the application.
    - *Reset:* No specific reset unless results alter application data.

- *Execution Steps:*

1. Request data export from the Settings or Profile page.
2. Verify that data is encrypted and secured, to be sent to the users profile email.
3. Check integrity and completeness of received data.

- *Execution Metrics:* Successful testing involves complete accurate data export and no security breaches reported.
Placeholder first attempt to execute this test case date/time: [To be filled out during testing]
Placeholder number of attempts to successfully execute this test case: [To be filled out during testing]
Placeholder list of software defects discovered by this test case: [To be filled out during testing]
Placeholder is this test case executed successfully? (yes/no): [To be filled out during testing]

CID: 41 - Title: Application Behavior Under Low Battery

- *Description:* Evaluate how the application behaves when the device is at low battery levels, ensuring speed and functionality is maintained, and no crashes or data loss happen.

- *Development Phase:* Stress Testing to simulate low battery scenarios to understand and gather data on how the application reacts to stressful conditions.

- *Test Goals and Measurement:* Our goal is that our application remains stable and saves data automatically when at low battery levels. Measurement by observing app behavior under set battery levels to ensure that performance is still good and no crashing results.

- Test Data/ Tools: Battery simulation tools, and manual device battery testing.
- Test Startup/Closedown/Reset Procedure:
    - *Startup:* Application running with typical user data and running basic operations.

- *Closedown:* Recharge device and restart the application to ensure normal conditions are present.
- *Reset:* No specific reset other than charging device back to normal battery operation.

- *Execution Steps:*

1. Simulate battery level performance at 20%, 10%, 5%, 1% while using the app.
2. Observe any changes in app performance, auto-save, notifications, data input/output.
3. Verify the app works as intended and data is secured and saved.

- *Execution Metrics:* Test is successful if the app functions correctly and secures data without the risk of loss across all battery levels.
Placeholder first attempt to execute this test case date/time: [To be filled out during testing]
Placeholder number of attempts to successfully execute this test case: [To be filled out during testing]
Placeholder list of software defects discovered by this test case: [To be filled out during testing]
Placeholder is this test case executed successfully? (yes/no): [To be filled out during testing]

CID: 42 - Title: Multi-Language Support Verification

- *Description:* Verify the application correctly supports multiple languages effectively and translates words accurately while maintaining our application's layout.

- *Development Phase:* Localization Testing as this phase assesses the application's ability to support various languages and cultural settings without functional or visual issues.

- *Test Goals and Measurement:* All app features work correctly in different languages, with UI elements properly aligned and text accurately translated.

Measurement by manual inspection and user/stakeholder feedback on language accuracy and reliability.

- Test Data/ Tools: Language packs, UI Language testing tools.
- Test Startup/Closedown/Reset Procedure:
    - *Startup:* Application configured in the default language (English) with options to select different languages.
    - *Closedown:* Reset application to default language settings and clear session data.
    - *Reset:* Revert any language-specific changes during testing.
    -

- *Execution Steps:*

1. Change app language in settings to a supported language
2. Navigate through the app verifying text and words are correctly displayed
3. Verify language support and display on each language to ensure layout and translation is correct.

- *Execution Metrics:* Success would be having all accurate translations while maintaining the correct layout throughout all application pages.
Placeholder first attempt to execute this test case date/time: [To be filled out during testing]
Placeholder number of attempts to successfully execute this test case: [To be filled out during testing]
Placeholder list of software defects discovered by this test case: [To be filled out during testing]
Placeholder is this test case executed successfully? (yes/no): [To be filled out during testing]

CID: 43 - Title: Device Compatibility Testing

- *Description:* Ensures the application functions correctly across different devices, maintaining usability and layout.

- *Development Phase:* Compatibility Testing, to focus on verifying the application's performance and presentation across a broad spectrum of environments.

- *Test Goals and Measurement:* Our goal is that the application displays and operates correctly on all targeted devices and OS versions. Measure through automated tests and manual checks through emulators like Android Studio to ensure consistency.

- Test Data/ Tools: Device and OS emulators, real devices testing
- Test Startup/Closedown/Reset Procedure:
    - *Startup:* Application installed and accessible across different devices
    - *Closedown:* Clear application data and log out from all devices
    - *Reset:* Reinstall or refresh the application to default settings for each device.

- *Execution Steps:*

1. Launch the app on different devices
2. Navigate through all app features, looking for layout and performance issues.
3. Test responsiveness and intakes of the app on different device screen sizes.

- *Execution Metrics:* Test is considered passed if the application runs as intended and seamlessly throughout all tested devices with no layout or performance issues.
Placeholder first attempt to execute this test case date/time: [To be filled out during testing]
Placeholder number of attempts to successfully execute this test case: [To be filled out during testing]
Placeholder list of software defects discovered by this test case: [To be filled out during testing]
Placeholder is this test case executed successfully? (yes/no): [To be filled out during testing]

CID: 44 - Title: Handling of Concurrent User Sessions

- *Description:* Test the app's ability to handle multiple sessions from the same user account, ensuring data consistency and session management.

- *Development Phase:* Load Testing. This phase evaluates the applications performance and reliability under our simulated conditions of having multiple devices under one user.

- *Test Goals and Measurement:* Goal is to validate the applications handling of sessions where the user is logged in through multiple devices, to ensure correct data sync, and avoid conflicts with data. Measurement by the absence of data loss, session errors, and feedback.

- Test Data/ Tools: Automated testing tools, Manual testing using multiple devices on the same account.
- Test Startup/Closedown/Reset Procedure:
    - *Startup:* User logged into the application from multiple devices
    - *Closedown:* Log out from all sessions and clear cache/mock data.

- *Execution Steps:*

1. Log into the same user account on multiple devices
2. Perform different operations at the same time on each device
3. Observe how the application handles updates, sync, user session management, and data upload.

- *Execution Metrics:* Successful if the application successfully is able to manage multiple devices using the same user account with successful data synchronization, updates, and management without data loss.
Placeholder first attempt to execute this test case date/time: [To be filled out during testing]
Placeholder number of attempts to successfully execute this test case: [To be filled out during testing]
Placeholder list of software defects discovered by this test case: [To be filled out during testing]
Placeholder is this test case executed successfully? (yes/no): [To be filled out during testing]

CID: 45 - Title: User Feedback Submission Process

- *Description:* Evaluate the process for users to submit feedback through the application, like form submission and response handling.

- *Development Phase:* Usability Testing, this phase focuses on the end user's and how they will interact with our system. This will be a good time to ensure our user feedback system is friendly and effective.

- *Test Goals and Measurement:* The objective is to ensure the feedback process is accessible and functional, with user feedback being stored and recorded accurately. Success is measured by accessibility, correctness, and user satisfaction with the process.

- Test Data/ Tools: Feedback form fields, server-side logging tools.
- Test Startup/Closedown/Reset Procedure:
    - *Startup:* Application configured to accept feedback, with test users ready to submit feedback.
    - *Closedown:* Securely store/delete feedback submissions; logout of application
    - *Reset:* Clear any mock feedback entries from the database to avoid clutter.

- *Execution Steps:*

1. Navigate to the feedback submission area in the application.
2. Fill in feedback form with various test issues to test validation
3. Submit the feedback and verify it is correctly received on the server.

- *Execution Metrics:* Test is successful if feedback is correctly submitted and received, with appropriate user notifications and submission status.
Placeholder first attempt to execute this test case date/time: [To be filled out during testing]
Placeholder number of attempts to successfully execute this test case: [To be filled out during testing]

Placeholder list of software defects discovered by this test case: [To be filled out during testing]
Placeholder is this test case executed successfully? (yes/no): [To be filled out during testing]

CID: 46 - Title: Recovery of Forgotten Password

- *Description:* Test the password recovery process for any user that has forgotten their account password, to ensure good security measures and user experience.

- *Development Phase:* Security Testing. This phase was chosen to ensure that the password recovery process protects user accounts against unauthorized access, and to make sure all data in the reset transaction is safe, which is how we will measure the success of the tests.

- *Test Goals and Measurement:* Our aim is to verify the security of the password recovery process, including how secure the data is. Success is measured by the process's ability to authenticate users, and how easy the password reset is for all users.

- Test Data/ Tools: Email system for recovery, testing tools for security verification.
- Test Startup/Closedown/Reset Procedure:
    - *Startup:* User accounts configured with accessible email addresses for recovery.
    - *Closedown:* Reset passwords for test accounts back to the original pre-test, and clear all temporary data.
    - *Reset:* Ensure all changes made during testing are successfully reverted to maintain security.

- *Execution Steps:*

1. User password recovery process on login page for a test account
2. Follow the recovery steps accurately, and use email verification

3. Reset the password and then attempt to login using the new set password.

- *Execution Metrics:* Successful recovery is measured by the users ability to reset their password successfully and access their account without any security issues.
Placeholder first attempt to execute this test case date/time: [To be filled out during testing]
Placeholder number of attempts to successfully execute this test case: [To be filled out during testing]
Placeholder list of software defects discovered by this test case: [To be filled out during testing]
Placeholder is this test case executed successfully? (yes/no): [To be filled out during testing]

CID: 47 - Title: Title: Notification for Goal Achievement

- *Description:* Verify that the application correctly notifies users upon achieving a set goal, like an application task or milestone.

- *Development Phase:* Functional Testing. This phase will allow us to test specific functions of goal achievement notifications to ensure that they work as intended with no errors.

- *Test Goals and Measurement:* Our goal is to validate the functionality of notifications for goal achievements, focusing on timeliness, accuracy and satisfaction with users. Success is measured by correct notification delivery, accuracy and user feedback.

- Test Data/ Tools: Test accounts with various goals set, notification logs.
- Test Startup/Closedown/Reset Procedure:
    - *Startup:* Configure application with test goals or milestones
    - *Closedown:* Clear any test data and log out of the application
    - *Reset:* Reset all manipulated goals or milestones to pre-test conditions.

- *Execution Steps:*

1. Perform actions that lead to a goal achievement
2. Verify goal achievement notification is displayed upon achieving the goal.
3. Check the relevance and accuracy of the notification.

- *Execution Metrics:* Success is the correct and timely delivery of notifications for each goal achievement.
Placeholder first attempt to execute this test case date/time: [To be filled out during testing]
Placeholder number of attempts to successfully execute this test case: [To be filled out during testing]
Placeholder list of software defects discovered by this test case: [To be filled out during testing]
Placeholder is this test case executed successfully? (yes/no): [To be filled out during testing]

CID: 48 - Title: Dark and Light Mode Functionality

- *Description:* Assess the applications support for dark and light mode, including UI consistency, readability, layout and preference settings.

- *Development Phase:* UI/UX Testing. This phase evaluates the application's user interface and user experience, which will align perfectly with the implementation and usability of our planned theme modes.

- *Test Goals and Measurement:* Our objective with this test is to ensure consistent and complete implementation of dark and light modes, with emphasis on user preference, performance and visual accessibility. Success is measured by the consistency of theme applications across app screens, ease of switching between modes, and user feedback.

- Test Data/ Tools: Manual Testing for visual inspection, UI Testing tools for theme consistency.
- Test Startup/Closedown/Reset Procedure:
    - *Startup:* Application with dark and light mode settings available
    - *Closedown:* Log out and reset application theme settings to default
    - *Reset:* Ensure the application is returned to its original theme mode before the testing began

- *Execution Steps:*

1. Navigate to app settings to switch between light and dark mode
2. Navigate through the app to verify UI consistency and readability in both modes.
3. Ensure the mode is saved after app closure or restart for both modes.

- *Execution Metrics:* Successful if both modes are correctly implemented across the app, with settings persisting across all sessions.
Placeholder first attempt to execute this test case date/time: [To be filled out during testing]
Placeholder number of attempts to successfully execute this test case: [To be filled out during testing]
Placeholder list of software defects discovered by this test case: [To be filled out during testing]
Placeholder is this test case executed successfully? (yes/no): [To be filled out during testing]

CID: 49 - Title: Wipe Storage Functionality

- *Description:* Verify the apps functionality to securely and completely wipe user data from the application storage on demand, and that the lost data cannot be recovered.

- *Development Phase:* Security Testing. This phase is crucial for assessing the application's ability to handle sensitive and secure data, making sure that the deletion of data is done in the safest way possible.

- *Test Goals and Measurement:* Our aim with this test is to verify that the wipe storage functionality effectively and securely removes all of the user data from the application and device storage, leaving no traces. Success is measured by making sure the data erased isn't able to be recovered, and complies with all data protection standards.

- Test Data/ Tools: Secure deletion tools, data recovery to verify wipe success.
- Test Startup/Closedown/Reset Procedure:

- *Startup:* Populate application storage with test data that we are able to identify for recovery attempts.
- *Closedown:* Securely wipe test data using the application's functionality.
- *Reset:* Verify the complete removal of test data, and ensure that storage is clean for the next tests to avoid wrong results.

- *Execution Steps:*

1. Initiate the wipe storage function in the application settings.
2. Verify that all app related data is completely removed from the device's storage.
3. Use data recovery tools to verify that the data cannot be recovered.

- *Execution Metrics:* Test is successful if no app data can be recovered post-wipe, to ensure a secure and accurate data deletion process.
Placeholder first attempt to execute this test case date/time: [To be filled out during testing]
Placeholder number of attempts to successfully execute this test case: [To be filled out during testing]
Placeholder list of software defects discovered by this test case: [To be filled out during testing]
Placeholder is this test case executed successfully? (yes/no): [To be filled out during testing]

CID: 50 - Title: Sign Out

- *Description:* Test the sign-out process, ensuring the user securely and successfully logs out of the application and clears any sensitive data.

- *Development Phase:* Security Testing, during this phase we can ensure that the sign-out process adheres to securities best practices, that protect user data and access to the application.

- *Test Goals and Measurement:* Sign-out process is secure and user-friendly, ending the user session and clearing all session data. Measurement by verifying no session or account data is accessible after sign-out has occurred.

- Test Data/ Tools: Manual Testing, Session monitoring and verification tools, and security assessment for checking data.
- Test Startup/Closedown/Reset Procedure:
    - *Startup:* User logged into the application, performing various actions that will create session data.
    - *Closedown:* Sign out of the application and close the application through the app.
    - *Reset:* Verify that no session data remains on the device or application cache.
    -

- *Execution Steps:*

1. Use the app and engage in different activities and screens
2. Sign out from the app using the provided sign out navigation
3. Attempt to access the app features and data to verify the session has ended.

- *Execution Metrics:* Successful sign-out is measured by the user's inability to perform any actions on the account without logging back in and ensuring all data is no longer accessible after a sign-out.
Placeholder first attempt to execute this test case date/time: [To be filled out during testing]
Placeholder number of attempts to successfully execute this test case: [To be filled out during testing]
Placeholder list of software defects discovered by this test case: [To be filled out during testing]
Placeholder is this test case executed successfully? (yes/no): [To be filled out during testing]

Top