1. Management Summary

The purpose of this document is to outline the test requirements and plans for the Android application known as Inspect, created by Binary Giant.
The testers involved will be the Binary Giant team – Michael, Jasmine, Conrad, and Elias. Each member will have different roles for testing which will move around to ensure each member is able to test a various number of scenarios throughout the development to ensure stability among multiple tests by different users.

2. Project Test Objectives

2.1. Security:

2.1.1. Screen Lock:

- Test Objective:
    - System inactivity that locks the screen, then unlocking the screen to resume template editing, saving, or inspection.
- Test Approach:
    - Test the results of what happens to the Inspect applications state when various devices lock when the app is in different states of use and ensure Inspect will be able to resume its previous state.
- Results and Deliverables:
    - When user unlocks device, Inspect resumes in same state it was in.
    - If Inspect detects it has been inactive for X minutes, it will prompt the user if they wish to continue (in case of no screen lock on device).
- Risks/Mitigation Strategies:
    - If the application goes into a “sleep” mode, it may not remember its state on waking if not saved properly and will need to be saved before sleep and reloaded on wake.
    - Multiple devices and multiple Android versions to test how sleep mode interacts with Inspect.

2.1.2. Storage:

- Test Objective:
    - Ensure the Inspect application is able to save templates and PDF exports to the Android device’s internal/external storage.
    - Ensure Inspect is able to load templates saved on the Android device for use.
- Test Approach:

    - Ensure exported files are correctly saved to a location on the Android device.
    - Verify that templates are imported/exported correctly with no missing or corrupt options.
- Results and Deliverables:
    - Templates are able to be saved and loaded to and from the Android device respectively.
    - PDF exports to the Android device are successful.
- Risks/Mitigation Strategies:
    - Data usage of the Android device requires sufficient storage to be readily available by the device.

2.2. Audit:

2.2.1. Logging:

- Test Objective:
    - Ensure logs are presentable and readable.
- Test Approach:
    - Test log file is able to be opened and read.
- Results and Deliverables:
    - Ensure errors presented are able to be used for future patches.

2.3. Performance

2.3.1. Start-up:

- Test Objective:
    - Application launch times.
- Test Approach:
    - Time how long it takes for Inspect to initialise.
- Results and Deliverables:
    - Fast enough so users will not spend time getting disgruntled and upset due to being forced to wait for Inspect to start up
- Risks/Mitigation Strategies:
    - Client can get irritated and cause them to go for another application/method to perform inspections

2.3.2. User Input:

- Test Objective:
    - Keyboard input, including loading times
- Test Approach:
    - Testing the speed on input from keyboard to be instantaneous updates to the inspection.
    - Time between user tapping on a keyboard input field and the keyboard appearing on the screen.
- Results and Deliverables:

        - Keyboard input should appear instantaneously in the inspection.
        - Load time for the keyboard to appear on the screen should be minimal (<1 second)

2.3.3. File export speed:

- Test Objective:
    - Speed for inspection export to PDF
- Test Approach:
    - Export various inspections to PDF format to test how long it takes for the export process.
- Results and Deliverables:
    - Minimal speed for exporting to PDF

2.3.4. Camera load times:

- Test Objective:
    - Camera open and response speed while user is performing an inspection.
- Test Approach:
    - Test opening the camera and ensure the load times are reasonable.
- Results and Deliverables:
    - No freeze's or delays with the camera starting up in Inspect.

2.3.5. Template load times:

- Test Objective:
    - Time to load template to begin an inspection.
- Test Approach:
    - Test opening the camera and ensure the load times are reasonable.
- Results and Deliverables:
    - No freeze's and minimal delays starting up an inspection in Inspect.

2.3.6. Double check export:

- Test Objective:
    - Ensure the system is able to export the PDF and verify to the user that information is correct before export.
- Test Approach:
    - Test an inspection and export a PDF.
- Results and Deliverables:
    - Ensure the PDF export is correctly structured and only exports when the user confirms.

2.4. Capacity

2.4.1. Data storage:

- Test Objective:

        - Ensure all data is appropriately sized and is stored correctly.
- Test Approach:
    - Output file sizes of templates, images, and PDFs.
- Results and Deliverables:
    - Low space used up to ensure user does not overfill their Android device with the Inspect application.

2.5. Availability

2.5.1. Storage:

- Test Objective:
    - Test various storage locations to find optimal storage location for user to easily find output files.
- Test Approach:
    - Test Android hierarchy browser in app to find exported items from the Inspect application.
- Results and Deliverables:
    - An easy to locate space on the Android device for the user to manually search through the file system and retrieve exported files, as well as an easy to access method from within the Inspect application.

2.6. Compatibility

2.6.1. Other system input:

- Test Objective:
    - Ensure camera and keyboard both work correctly.
- Test Approach:
    - Ensure inputs from Android devices keyboard and camera are inputted into the Inspect application correctly
- Results and Deliverables:
    - Fully functional keyboard and camera integration with the Inspect application

2.6.2. Interface accessibility for colour blindness:

- Test Objective:
    - To ensure that the correct colour combinations are employed to enhance interface accessibility for those suffering from colour blindness.
- Test Approach:
    - Research the various types of colour blindness, determine a spectrum of colours visible to those afflicted by it and find an appropriate combination of colours to optimise visibility.
- Results and Deliverables:

- An appropriate palette/combination of colours to be used for the interface that are visible to the majority of users who are colour-blind.

3. Test strategy

3.1. Product risk analysis (see template)

Risk	Classification
Data loss	Storage, Security, Stability
Application crash	Stability, Compatibility
Slow User Interface	Performance

3.2. Test Levels – Test Strategy

Test level	Goal
Unit Testing	Ensure methods within classes give expected output and meet user specifications.
Component Testing	Ensure that two components of the same module behave in an expected manner when they interact.
Integration Testing	Ensure that two modules behave and function as expected after integration.
System Integration Testing	Ensure that all related systems maintain data integrity and can operate in coordination with other systems.
User Acceptance Testing	Ensure that Use Cases can be executed and the system behaves and functions as expected in real world scenarios.

4. Infrastructure

4.1. Test environments

Test level	Test environments
Unit Testing	Android Studio
Integration Testing	Android Studio
User Acceptance Testing	Emulators and Physical Devices

4.2. Test Tools

Test level	Test Tools
Unit Testing	J-Unit
Integration Testing	Mockito and Espresso
User Acceptance Testing	Emulators and Physical Devices.

5. Test management

5.1. Test process management

Test level	Guidelines
Unit Testing	Is done by the author of the code before merge and commit to master.

Integration Testing	Will be done by all team members for all components being integrated during each Elaboration Phase.
User Acceptance Testing	Will be done internally by all team members in the last Elaboration Iteration and for all Construction Phases. Externally this will occur during the Transition Phases.

5.2. Defects Procedure

For the registration and maintenance of defects the following tool will be used: Finotes.

- Defect is identified
    - Check defect list to see if defect has already been identified.
    - If defect is already present in the list, team member will fill in any missing details. No further action is required.
    - If the identified defect is not present, add defect to the defect list and complete all necessary fields.
    - Assign a team member. Ideally it will be the author of the code that the defect appears to be in.
- Investigation by assigned team member:
    - Reproduce issue.
    - Determine source of defect or
    - Determine it is not a defect and mark as rejected.
- Resolution:
    - Repair the issue.
    - Mark as “To be verified”.
- Verification by Team member who initially reported the defect:
    - Verifies defect is resolved.
    - If it is resolved mark as closed, otherwise reopen defect.