Elaboration Phase Status Assessment

1. Assessment against Objectives of the Elaboration Phase

1.1 What were we trying to achieve?

The aim of the project is to create a mobile application that streamlines any kind of inspection. It will allow the creation of forms and allow the user to fill these within the application. These forms can be filled out during the inspection and then sent to clients after they are completed via the share function. This is embodied in the completed Vision Document.

We understand the main functional requirements of the project which are:

- Create Form Templates
- Modify Form Templates
- Loading and Saving Form Templates
- User fill forms (during inspection)
- Saving User Form
- Export to pdf
- Distribute via email

We understand the main Non-Functional requirements of the project which are:

- Usability
    - Applying good design practices and concepts to create a familiarity with the application to optimise usability.
- Reliability
    - The system will be designed to work offline with the majority of files saving directly to the device. However, the share feature within the application will require internet connection.
- Performance
    - Needs to respond in a timely manner to the user. Any delay in loading should be displayed to the user via prompts.
- Maintainability
    - The system will conform to a modular architecture, by breaking the architecture into smaller components. This will make the code easier to read and understand.
- Capacity
    - Only one device is required to use the system. The application will not require much storage to be installed. However, device storage will limit the amount of templates that can be saved.

- Compatibility
    - The target hardware is a mobile phone with android OS installed. The device will need to be able to type and select input. It will also require a rear facing camera. PDF has been choice as the output format

1.2 What have we achieved?

We have achieved several aim within the project. We have used our IDE choice, Android Studio, to create a proof of functionality through this semester. The user is able to:

- Create a template
- Load and modify the template
- Fill in the template
- Access their devices file system to store and retrieve templates
- Take photos using their device's camera, which can be placed into the inspection form
- Select photos from gallery on their device, which can be placed into the inspection form
- Share their template to email etc. through android's share services

1.3 What still needs to be done?

Even though we have completed may of our main objectives, we still need too:

- Fix two bugs
    - Screen Rotation issues, which involves implementing multiple different libraries to properly fix
    - Blank pages added to template and then previewed, crashing the application
- Create a more user friendly approach to the interface
- Better separate template editor and inspection sides of the application

2. Deliverables

2.1 Android Application

2.1.1 Conversion Manager

Wasn't needed as Android was able to handle that process with a view adapter.

2.1.2 Log Manager

There were some problems initially in ensuring the application was successfully saving and reading log files via the Log Manager. The method itself had to be revised from previous iterations as the buffered writer employed was not creating the log files in the directory. Even though we tried to pinpoint the issue through research and online tutorials and came up short in identifying the issue with the buffered writer, an alternate method in outputFileMethod was employed, which managed to successfully read, save and create the file. Once we fixed this issue, the Log Manager had no

other issues. Catching any issues and reporting them, as well as updating the user's current status to assist with tracking was simple enough.

2.1.3 PDF Export

The ViewPrintAdapter handles the PDF export by accepting a view, rendering said view as a bitmap, and placing each bitmap as a page in a PDF document.
There was some minor issues around formatting of views, and as we move forward we need to ensure that the view formatting is respected. One issue left unresolved is that the Android Print to Pdf wants to place the bitmap centrally on the page, but as the app progresses this will be solved either directly in the PDF printer or with an element weight system that pads the image to achieve the desired alignment.

2.1.4 Template Editor

Errors found late in development cause massive issues for the scope of the project. When the device was rotated to landscape the view would reset. The only fix for this issue found was to implement rooms, live data and view model libraries provided by android. These would be used in such as way that data wasn't being held by the views but fed into them. To implement this look longer than intended as majority of tutorials are in Kotlin not Java.

2.1.5 File manager

The file manager required integration with the Storage Access Framework (SAF) which was an issue in itself. Working with Android for the first time and integrating this system had proven quite challenging as the documentation was limited and had to rely on forum posts for some errors that were showing up. Once we had found out the issues and required abstract classes not inside an activity we were able to get the system working correctly. After the SAF was able to open, grabbing the file URI's was simple.

2.1.6 File Share Manager

During the initial phases, there were some hurdles to overcome in ensuring certain aspects of the class within the Android Manifest were registered. This was remedied through a combination of extensive research online, assistance from other team members and correctly updating the File Provider permissions within it. Other areas where issues arose included a few of the methods that needed to call variables, which in turn needed to be set up by creating a filepaths .xml file with TextView. Trying to run these components proved bothersome as the emulator either kept shutting down or failed to load certain aspects of the application for some team members but appeared to run fine for others. This was easily rectified by creating a new emulator and running the application on it.

2.1.6 Template Manager

This was implemented using the file manager.

2.1.7 Photo Manager and Camera Integration

There were initial issues regarding the File Provider, mainly in returning the correct values for the URI's in creating a file (and directory) and sharing it. This also lead to other issues such as being unable to take a photo and save it to the photo gallery. We ended up working on alternate implementations, ensuring null values weren't being returned for the URI's, which proved to be part of the initial problem. Through a process of trial and error and extensive research, we were able to correct these errors so that files could be shared and photos saved to the photo gallery.

2.1.8 Inspector

Due to issues with Template Editor, The editor and inspection functionality was moved to the same class. In editor mode you can add elements and fully edit them and in inspection mode this editing functionality is omitted and labels are read only leaving only answer fields to be editable by the user/inspector.

2.2 Architectural Notebook

The Architectural Notebook took longer than expected. Initially there was some confusion regarding the Architectural Mechanisms portion, but through dialogue with team members, they were able to be discerned by carefully reviewing the Non-Functional Requirements.

There was also a degree of uncertainty when approaching Component Diagram for the Architectural Notebook in presenting in the architectural view of the system and whether it correctly illustrates the system as viewed through the proposed three-tier architecture. This also extended to operational views that were based on the Component Diagram, but other team members were able to review these items and came to the conclusion that the Notebook components formed a solid foundation for the other documentation. We decided that the Notebook's contents were malleable at this stage and were expected to be revised in later phases/iterations.

With regards to the diagrams that would communicate physical/logical views of the system, the team had some trouble deciding on the correct domain model diagram to be included out of the two initially created, or if some combination of the two would be desirable for this portion of the Notebook. One of the diagrams functioned better as a class diagram because of the logical views presented between the key abstractions such as Inspector, Fault, Inspection Site, etc. whereas the other provided insight into the conceptual layer that is characteristic of domain models.

We eventually decided on the latter as the best representation of these physical/logical views because it already incorporated class diagrams to illustrate this. We also wanted to avoid further delaying other documents which depended on it. Combining the two domain model diagrams would have proved to be quite cumbersome and ultimately redundant.

2.3 Risk Assessment

No Issues

2.4 Master Test Plan

No Issues

2.5 User Acceptance Testing

No Issues

2.6 Project plan

No Issues

2.7 Use Case Models

No Issues

2.8 Vision Document

No Issues

2.10 Test Report

No Issues

2.11 Executable Architecture

No Issues

3. General Issues

3.1 Skill and knowledge

Skill and knowledge still causing major issues in the time taken for development. Research is taking up the bulk of assigned time per week. As time goes on this will improve. This issue is ongoing as new work items require more research.

3.2 Productivity Issues - Balancing External and Internal Pressures

While this is inevitable, good communication, coordination and flexibility of all team member will be used to mitigate the issue.

3.3 Lack of understanding how android object handle data

During implementation of the template editor, an error was found when rotating the device. Due to a lack of understanding of how android handled data sorted within views, this issue then affected other parts of the application.Information displayed in the views only last as long as the view was active. Meaning when the device was rotated or a view was changed the data displayed would be lost.

Issue is ongoing and is on the way of being completed.

3.4 Lack of Tutorials for Java on Newer Concepts

A lot of current tutorials are written for Koltin now rather than Java, it can be difficult to find helpful tutorials for newer libraries and concepts written in Java.

4. Risks

4.1 Overall Application Architecture

The only viable way to fix the data lost when views are changed is to implement the 3 libraries, rooms, live data and view models. If this strategy can not be implemented when intended the project might have to live with this data lost.

5 Summary – Overall Project Progress

The project has moved along with few issues arising. These were solved by the team as we expected to hit road bumps due to the team's experience developing within the Android system which caused bugs to be created and fixed throughout the elaboration phase. The functionality of the application is there but can be improved to increase the usability. While bugs that cause the application to crash have mostly been rooted out, there is a requirement for more in depth testing to ensure maximum usability within the application.