IntelliWork

ECE1778 Final Report

Due Date: December 14, 2016
Word Count: 2020 + 433 = 2453
Specialist: Jiawei Du (998711327)
Programmers: Louis Tsui (998995057), JiaRen Bai (998884685)

Table of Contents

1. Introduction

Chemical manufacturing plants are complex systems that contain various types of machines. These machines act together to form a unit functionality, and one single machine failure can have catastrophic consequences for the system as a whole. To mitigate risks, machinery operators, who are directly involved in the production processes, conduct unscheduled inspections aside from routine tasks. They are required to check on the physical conditions of equipment on the plant floor on a regular basis to ensure the plant is running normally. According to the interview with operators from a toner manufacturing plant, the majority of machine failures are identified during unscheduled inspections. Therefore, assisting them to identify symptoms of equipment in distress conditions and to take actions to minimize those impacts, is essential to improving and sustaining asset reliability.

Unscheduled inspection is a critical but also challenging task. Currently, operators have limited access to key operating data when they are conducting unscheduled inspections on the plant floor. The lack of information access inhibits them in making informative decisions, and as a result, operators commonly experience high mental workload especially during emergent conditions. If an abnormal symptom is identified, operators report the incident and request assistance verbally via radio. However, verbal communication is not effective in describing equipment conditions as compared to using photos and videos. In addition, all documentation of inspection outcomes are completed manually. Not only are hand-written reports prone to human errors, the manual processes are also labour intensive and time-consuming. The unstructured inspection process poses high execution variability, exposing the system to potential hazards of catastrophic consequences. Given these challenges, both management team and operators agree that there is a need for an appropriate tool to support inspection related tasks.

The goal of the app, IntelliWork, is to assist plant operators to inspect production processes in a more systematic and efficient way. The app will provide a guideline that standardizes unscheduled inspection processes. It will also provide functionalities that help operators to identify, communicate, and document incidents during inspections.

2. Statement of Functionality & Screenshots from App

The app consists of several screens that contain various functionalities to support unscheduled inspections. End-users of this app are mainly machinery operators, team leads, and engineers. Screenshots are taken from IntelliWork's video demo.

The menu screen (Figure 2.1) lets users choose types of inspection tasks to complete. The "Quick Start" button contains major core functionalities. Upon selecting "Quick Start", users see three tabs.

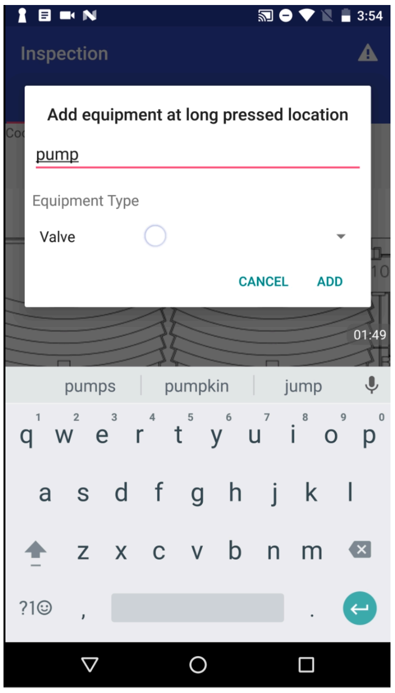


Figure 2.1 Menu screen

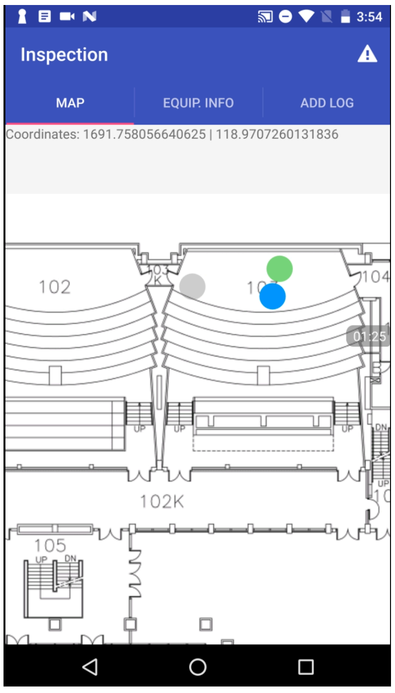


Figure 2.2 Map tab

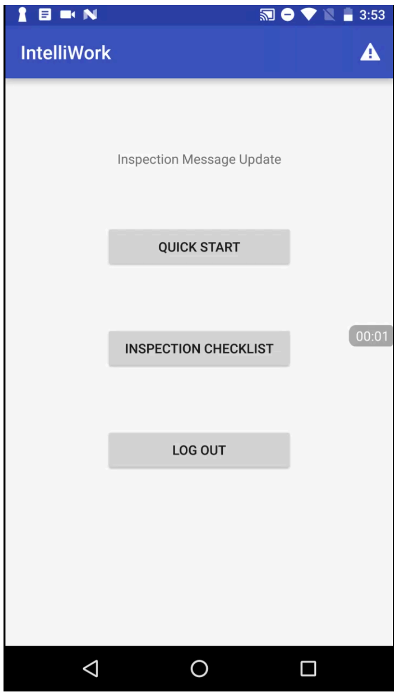
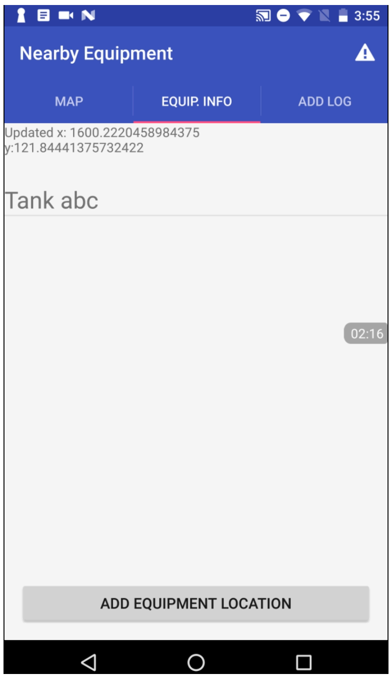


Figure 2.3 Adding equipment

The Map tab (Figure 2.2) displays the building floor plan, where the user is located. The map view can be zoomed in and out by pinching and panned using short swipes. The map also draws several dots. A single blue dot tracks the user's position and movement. The other dots represent equipment: equipment nearby users is highlighted in green while equipment out of range is shown in grey. In this demo, we considered and set "nearby" to be within five meters of the user's position.

Users can also add equipment locations on the map to match with any newly installed or relocated equipment on the floor. They do so by performing a long press on the area where the equipment is located, and this brings up a dialog that requests information inputs (Figure 2.3).

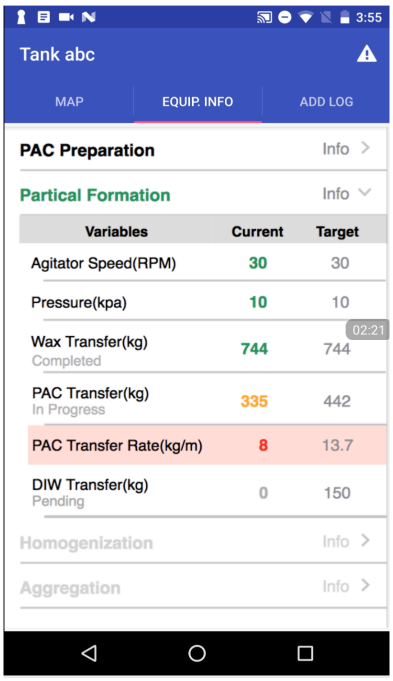
The Equipment Info tab (Figure 2.4) displays a list of equipment nearby the user (equipment is displayed in green dots). Tapping on a list entry brings up the particular equipment’s data (currently only a placeholder image is displayed) (Figure 2.5).



Updated x: 1600.2220458984375
y:121.84441375732422

Tank abc

Figure 2.4 Equipment info tab



PAC Preparation — Info

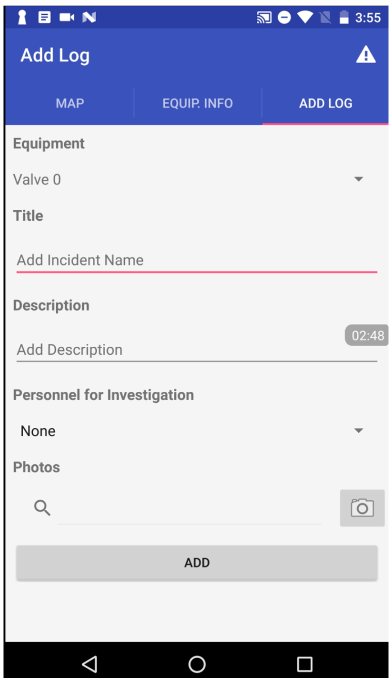
Partical Formation — Info

Variables	Current	Target
Agitator Speed(RPM)	30	30
Pressure(kpa)	10	10
Wax Transfer(kg) Completed	744	744
PAC Transfer(kg) In Progress	335	442
PAC Transfer Rate(kg/m)	8	13.7
DIW Transfer(kg) Pending	0	150

Homogenization — Info

Aggregation — Info

Figure 2.5 Equipment details



Equipment: Valve 0

Title: Add Incident Name

Description: Add Description

Personnel for Investigation: None

Photos

Figure 2.6 Add log tab

The Add Log tab (Figure 2.6) contains a form to fill out incident details: the equipment related to the incident, title, description, personnel to be involved, and the option to take a photo for additional information (Figure 2.7). Upon pressing the “Add Log” button, the incident is uploaded to a remote server and made available for other users. The corresponding equipment of this incident is marked with a wrench icon on the map (Figure 8).

Upon selecting “Inspection Checklist”, the user sees various inspection checklist templates based on floor plan locations (Figure 2.9). Each checklist template contains a list of items (questions and tasks) to help users summarize inspection outcomes systematically (Figure 2.10).

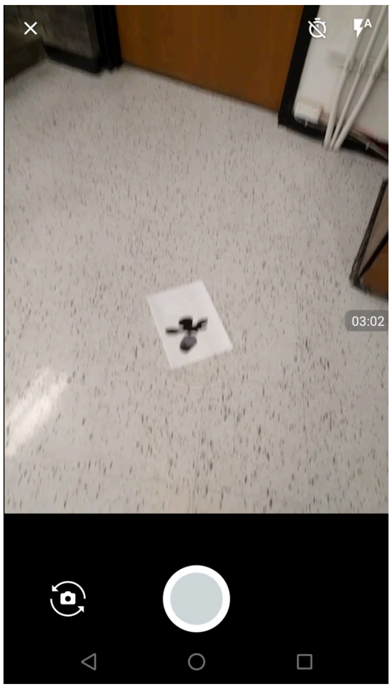
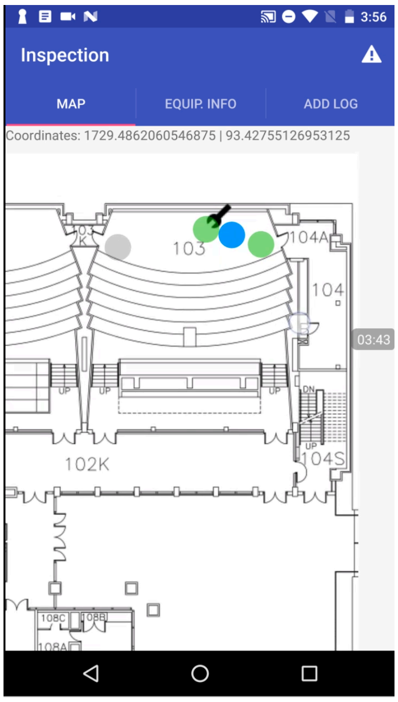
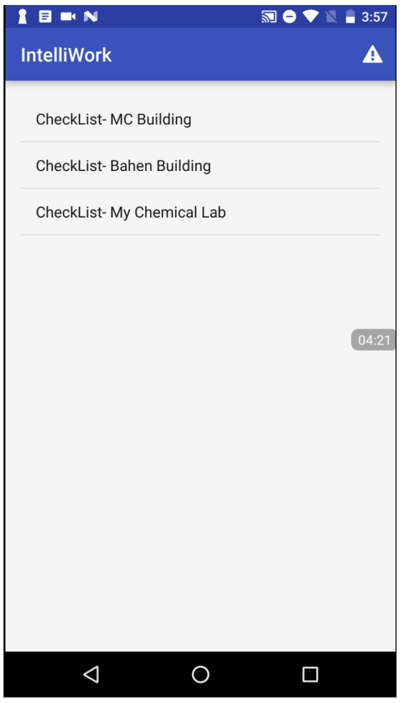


Figure 2. 7 Photo of incident



Coordinates: 1729.4862060546875 | 93.42755126953125

Figure 2.8 Incident on map



CheckList- MC Building

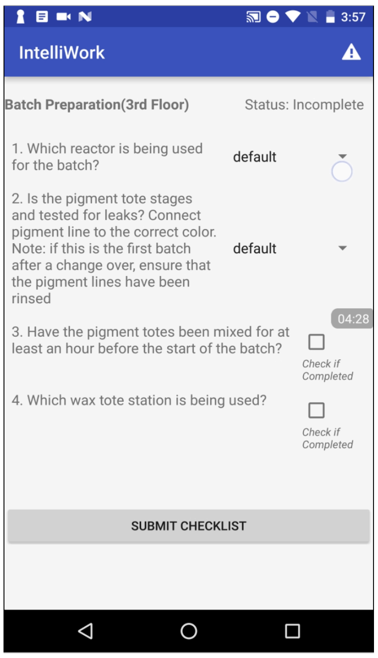
CheckList- Bahen Building

CheckList- My Chemical Lab

Figure 2.9 List of checklists

The warning icon on the menu bar represents the “Incident Logbook” that contains three tabs.

The “Unresolved” tab contains unaddressed incidents (Figure 2.11). Each incident on this tab contains a summary of details, a timestamp, and a popup menu with additional options. The popup menu has two more options: “Mark as Done” flags the incident as completed and “Get Details” brings up a screen titled “Full Detail” which shows a standardized incident report(Figure 2.12).



Batch Preparation(3rd Floor) Status: Incomplete

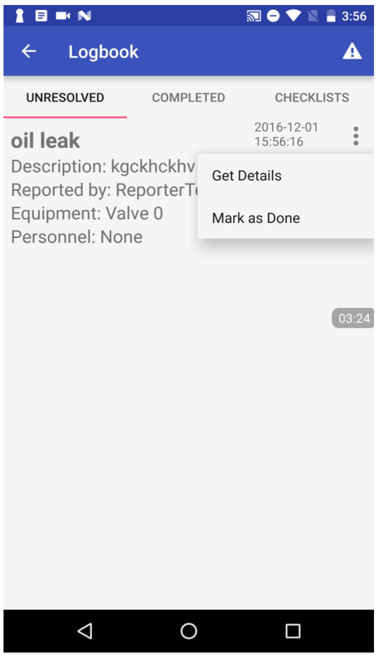
1. Which reactor is being used for the batch? default

2. Is the pigment tote stages and tested for leaks? Connect pigment line to the correct color. Note: if this is the first batch after a change over, ensure that the pigment lines have been rinsed default

3. Have the pigment totes been mixed for at least an hour before the start of the batch? ☐ Check if Completed

4. Which wax tote station is being used? ☐ Check if Completed

Figure 2.10 Checklist



oil leak 2016-12-01 15:56:16

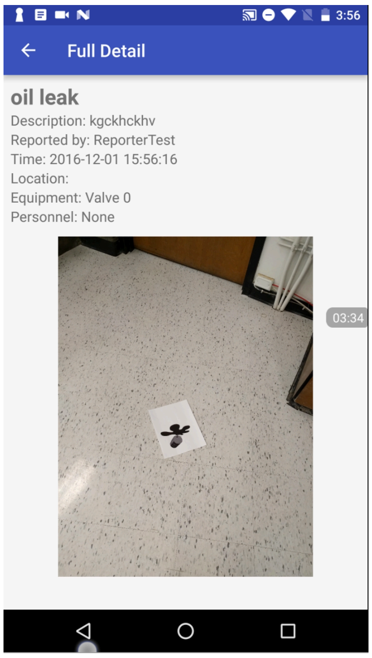
Description: kgckhckhv

Reported by: ReporterT

Equipment: Valve 0

Personnel: None

Figure 2.11 Unresolved tab



oil leak

Description: kgckhckhv

Reported by: ReporterTest

Time: 2016-12-01 15:56:16

Location:

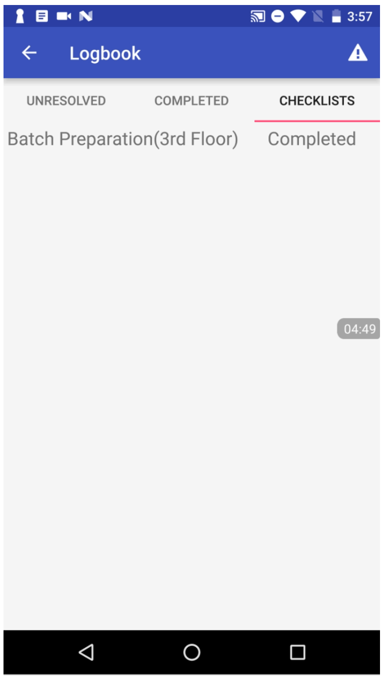
Equipment: Valve 0

Personnel: None

Figure 2.12 Incident details

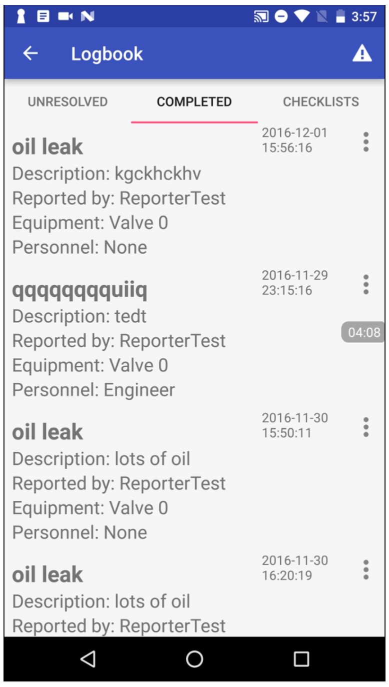
The "Completed" tab contains incidents that have been resolved via the aforementioned "Mark as Done" button. The popup menu contains a "Revert as unresolved" button instead of "Mark as Done". This feature allows the user to undo changes if an incident is marked as done by mistake.

Finally the "Checklist" tab contains a historical record of checklists that have been submitted. Each list item shows the type of the checklist and its status (whether completed or not).



Batch Preparation(3rd Floor) Completed

Figure 2.13 Completed tab



oil leak
2016-12-01 15:56:16
Description: kgckhckhv
Reported by: ReporterTest
Equipment: Valve 0
Personnel: None

qqqqqqqquiiq
2016-11-29 23:15:16
Description: tedt
Reported by: ReporterTest
Equipment: Valve 0
Personnel: Engineer

oil leak
2016-11-30 15:50:11
Description: lots of oil
Reported by: ReporterTest
Equipment: Valve 0
Personnel: None

oil leak
2016-11-30 16:20:19
Description: lots of oil
Reported by: ReporterTest

Figure 2.14 Checklist tab

3. Overall Design

The **block diagram** is divided up into the application front end and the server back end. The front end (Android Application End) is responsible for mapping functionalities and the back end handles data storage.

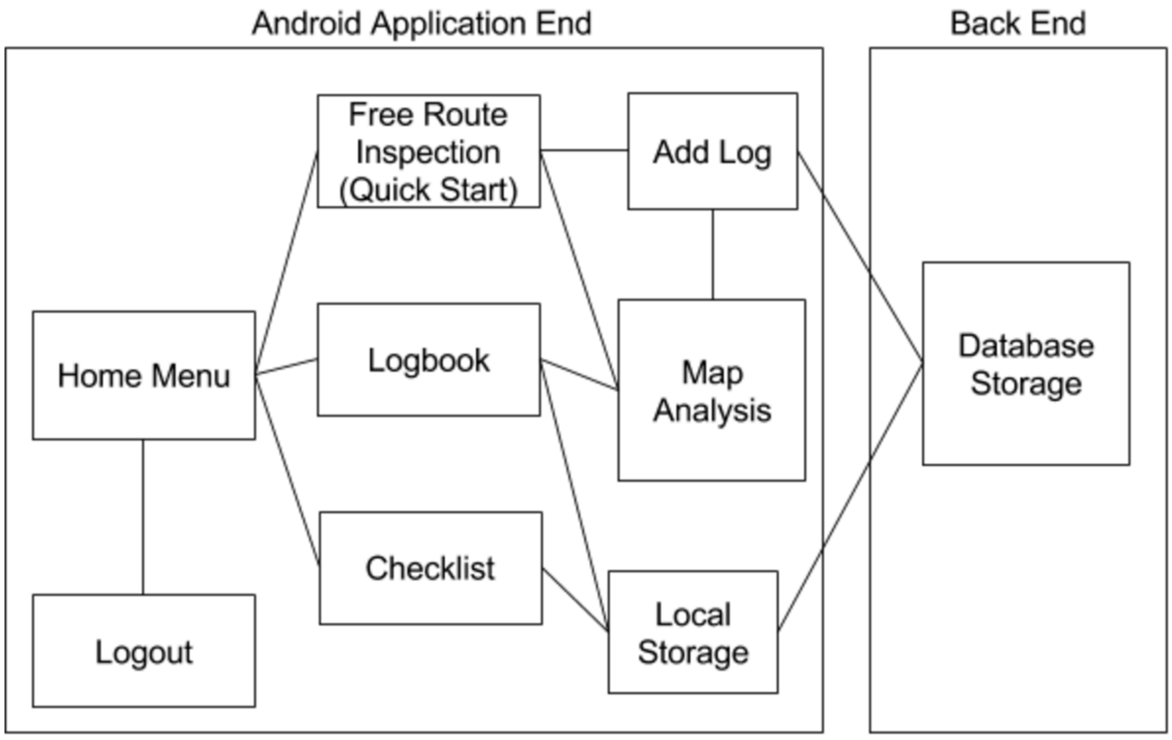


Figure 3.1 Block diagram representing software structure

The **Home Menu** block consists of several buttons that allow access to different app features.

The **Logout** block handles cleaning up various temporary code objects created during the application run before exiting.

The **Free Route Inspection** block displays a floor plan of the building the user is in and provides a list of equipment. It relies on the Map Analysis block to determine nearby equipment and have their details accessible.

The **Add Log** block contains the standardized form to document details of an incident, which then will be sent to the remote server. After an incident is created, the Add Log module communicates to the Map Analysis block to update icons; this means the equipment with the incident reported will have a wrench icon added on the map.

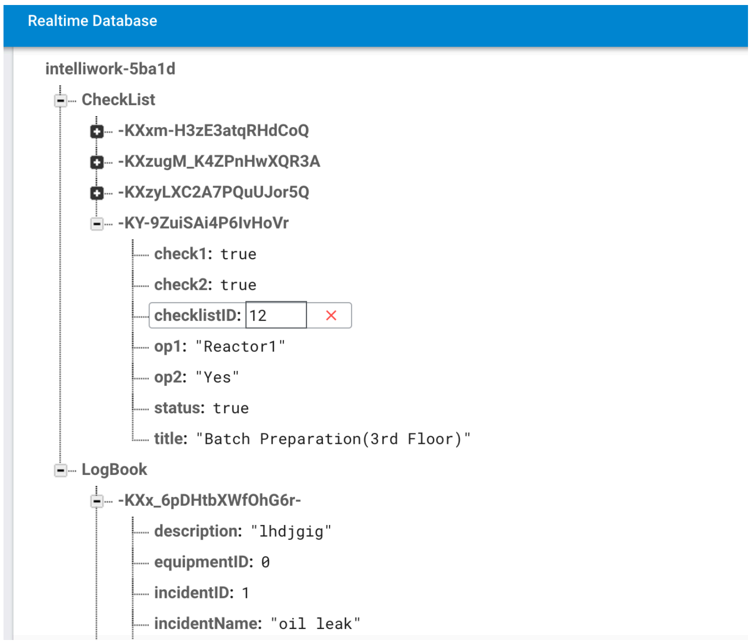
The **Logbook** block records and displays inspection results, which includes incidents and checklists. This module synchronizes with the Map Analysis whenever there is a data change. For example, after an incident is set to "completed", the Logbook notifies the Map Analysis to erase the corresponding wrench icon from the map. Likewise, when an incident is reset to “unresolved”, the Map Analysis is notified to redraw the wrench icon for that equipment on the map.

The **Map Analysis** block contains the logic to analyze the user’s position and display information in their surroundings. Equipment list detection, for example, is done by consistently computing the distances between the current position and the equipment location. The equipment within the range is then filtered out and displayed. Other examples include changing equipment dot colours or drawing a wrench icon (unresolved incident) on an equipment dot. We relied on two software libraries to implement this functionality: IndoorAtlas and David Morrissey’s Subsampling Scale Image View (SSIV). IndoorAtlas is a navigation technology based on geomagnetic positioning and a mapping process is required to set up the system. SSIV is to support displaying the map.

The **Checklist** block provides users a systematic way of inspections. The module consists checklist templates; each template has several fields such as questions and tasks to complete.

The **Local Storage** block represents equipment, incidents, and checklists that are all saved as Realm database objects. The module extracts necessary information by querying Realm based on users’ requests.

The **Database Storage** block is a Firebase Database (Figure 3.2) that backs up information from the local Realm Database; this way, data is made available to multiple users. Upon any data change, the update is saved both locally and remotely. So, all users will have the latest data access.



```
Realtime Database

intelliwork-5ba1d
  CheckList
    -KXxm-H3zE3atqRHdCoQ
    -KXzugM_K4ZPnHwXQR3A
    -KXzyLXC2A7PQuUJor5Q
    -KY-9ZuiSAi4P6lvHoVr
      check1: true
      check2: true
      checklistID: 12
      op1: "Reactor1"
      op2: "Yes"
      status: true
      title: "Batch Preparation(3rd Floor)"
  LogBook
    -KXx_6pDHtbXWfOhG6r-
      description: "lhdjgig"
      equipmentID: 0
      incidentID: 1
      incidentName: "oil leak"
```

Figure 3.2 Remote Firebase server database schema

4. Reflection

Technical

First and foremost, we learned how to build a sizeable Android project and to integrate third-party libraries into an app of our own. We gained experience with looking up online documentation and resources when implementing features and debugging issues. Utilizing third-party software libraries sped up development progress and freely choosing those libraries was an opportunity typically not experienced in previous course assignments.

We should have carefully planned out the database design beforehand. Later additional schema changes were rather time-consuming as we tried to merge those changes with existing infrastructures; the process also created several bugs to fix. It would be a good practice to design a database scheme beforehand that considers all potential app components and synchronization mechanisms. This way, programmers could also implement interfaces to facilitate synchronization process between local and remote database storage.

Non-technical

First, we learned how to narrow down the app scope, i.e. what the app does and why. This became a clear guideline when defining use cases of an indoor positioning technology. We also learned how to make a pitch when demonstrating our app.

We should have visited the plant more frequently to gather feedback. From our plant revisit in November, we received highly positive comments and valuable feedback for future improvements. But due to time constraints, we were only able to implement the checklist functionality. Other suggestions are discussed in the section "Future Work".

5. Contribution by Each Group Member

Collaboration by all team members:

- Discussed the contents and slides for each presentation and practiced with other members
- IndoorAtlas setup: floor-plan mapping

Individual Contribution:

Louis Tsui

- Implemented the majority of the user interface (UI)
    - Logbook and Quick Start activities
    - Tabs
    - Associated fragments (except the Checklist)

    - Dialog fragments
    - Action bar
- Connected UI components to Realm objects which involved querying Realm and accessing the retrieved objects' attributes
- Developed the core mapping functionality (drawing of the map and blue dot of the user) which was heavily based on the examples provided by IndoorAtlas
- Led the debugging effort and part of this process involved searching for documentation online to see what design patterns might resolve our issues
- Coordinated with JiaRen to manage a git repository for development
- Testing the app in the presentation environment was done collaboratively

JiaRen Bai
- Developed equipment list detections under the Map Analysis module and the Checklist module
- Developed equipment storage and data synchronization mechanisms under the Local Storage and Database Storage modules
- Coded equipment and incident icons, which would be concurrently displayed on the map
- Debugged code modules and tested IntelliWork to ensure reliability

Jiawei Du
Project Development:
- Defined main objectives of the app based on interviews with operators (end users) during the site visit to a toner manufacturing plant
- Came up with the ideas for indoor positioning; defined use cases of indoor positioning system in a chemical plant setting
- Designed basic app functionalities
- Designed user interfaces and created mockups of a high-fidelity prototype using Moqups.

Communication:
- Explained the context of my research field to help programmers better understand the motivations and goals of the app
- Revisited the plant to presented our prototype to operators and collect their feedbacks

6. Specialist Context

I am from the Cognitive Engineering Laboratory at the University of Toronto. My research group, in partnership with ABB Canada, seeks to better understand the application of adaptive automation to complex systems. The goal of my research project is to support inspection-related tasks through adaptive mobile application. The ECE1778 project contributes directly to my research project because the prototype our team designed and developed in this course can be used to examine the effectiveness of adaptive automation in process industries.

This prototype, Intelliwork, was designed by using Feigh et al's Adaptive System framework [1]. According to the framework, an adaptive system allows both users and the system to change its level of automation depending on the context. An adaptive system contains two parts: 1) an adaptation mechanism that categorizes ways a system can adapt its behaviours and 2) a trigger mechanism that describes how adaptive systems sense the current situation and decide when and how to change. My research aims to investigate *Spatio-Temporal Trigger on Information Content Adaptation*, i.e. how time and location can be used as adaptation triggers to change information that system presents to the user. Several simple adaptive features have been implemented in the design. For example, the color of equipment markers (*information content*) will be adapted according to user's position (*spatio trigger*), and only information of nearby equipment (*spatio trigger)* is accessible from the "nearby equipment list" (*information content*). By highlighting the nearby equipment that is more likely to contain relevant information to user's tasks, the adaptive application is expected to increase task efficiency by reducing time that the user requires to process information. As a result, users are expected to improve their performance and experience less mental workload.

From industrial perspectives, the successful deployment of this adaptive mechanism has the potential to fundamentally change the way chemical plants are operated. In the past, operating data is only accessible in the centralized control room, meaning that the entire production process can only be monitored and controlled remotely from the control room. Disconnected from the physical working environment, operators often suffer from high cognitive workload because they have to reply on abstract information on the display to make sense how well the plant is functioning. In the advent of mobile and sensing technology, mobile devices present a compelling opportunity to break the spatial segregation between the plant floor and control room by delivering system data to the point of work. In particular, adaptive mobile application will be exceptionally useful because it can work together with human operators as an intelligent agent to help them complete the tasks more efficiently.

7. Future Work

Technical Work

The team would like to enhance the code module that displays up-to-date equipment data in terms of tables and graphs based on equipment types. As plant operators have suggested, mobile equipment information access would be greatly appreciated, so this feature can facilitate their inspection process.
The team also wants to include more checklist templates that will be designed based on users' requirements; this feature may improve users' inspection efficiency.

Currently, the synchronization process involves converting data from Realm to Firebase databases and is handled by loosely structured code. We would like to refactor the code to implement an interface that communicates between the two technologies. Doing so would make it easier to organize future code in case of any database schema changes. Extending the code base would be simpler and less prone to bugs.

Research Work

This app will be used as the prototype in specialist's research experiments to examine several design concepts of adaptive automation in the context of process industries.

8. Course Website Permissions

Video of final presentation: Yes
Report: Yes
Source code: No, the research sponsor ABB would like to keep code closed-source.

9. Reference

[1] Feigh, K. M., Dorneich, M. C., & Hayes, C. C. (2012). Toward a Characterization of Adaptive Systems A Framework for Researchers and System Designers. *Human Factors*, 54(6), 1008-1024.