Riverwatch Android Application

Technical Design Document

TABLE OF CONTENTS

1 Introduction

1.1 Purpose

This document is to provide future developers of the Android Riverwatch App with information regarding the high level architecture of the application, the sequence of events per user action, and the sequence of events comprising the “Out of Area Caching”. Also contained is a high level diagram using images of the existing layout and the views shown per user action.

The document is provided as a guide for future developers of this application, and must not be used as an exact guide to reproduce the application exactly. Team Synergy2 is not liable for any costs or damages incurred by following this document, and as such it is at the developers’ discretion as to how this document is to be used.

It is advisable for any developer using this document to update changes to the software.

1.2 Scope

- Development of an Android app with the following functionalities:
    - allows users to take photos of polluted rivers and submit them to the WaiNZ website, along with a description of the incident and GPS coordinates of the area.
    - Caching data for submission when the user is outside mobile coverage range.
    - Adding image names to submitted images.
    - Authenticating the user of the app
- Testing of the Android App
- Source Code and documentation repository locations

2 System Overview

2.1 System Architecture

fig1. The following diagram gives a high level view of how the app interacts with the server, and the device's internal storage.

High Level Architecture Diagram of the Riverwatch Android Application

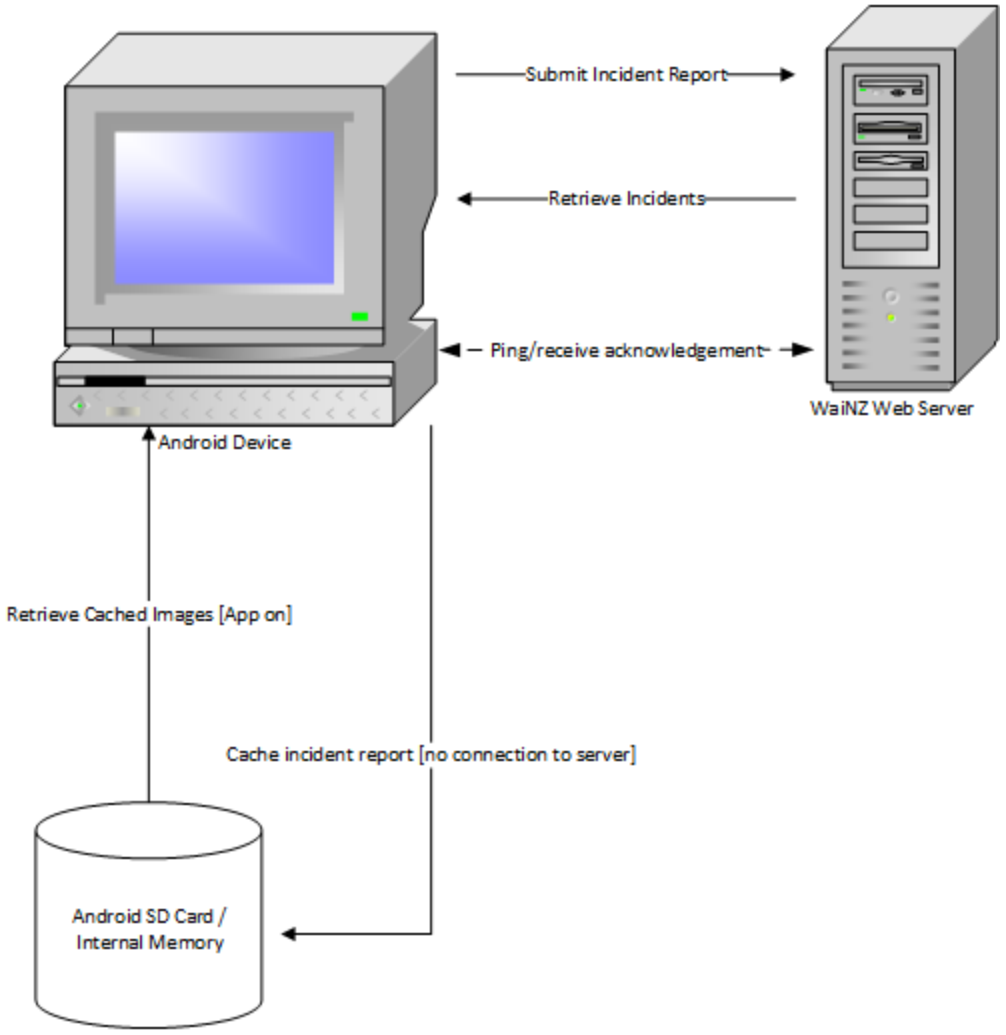
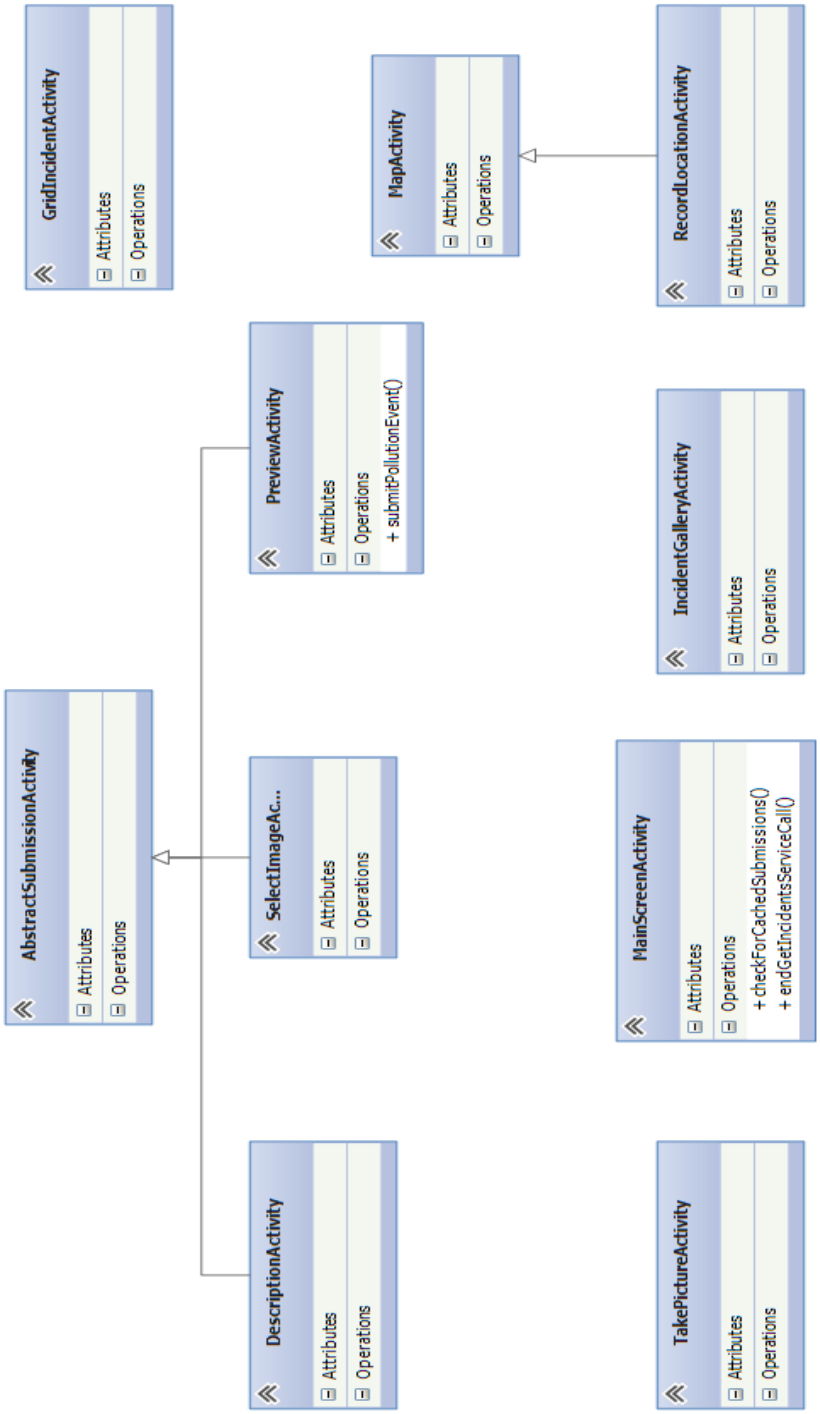


fig2. Class Diagram showing class association, and where the incident submission/out of area caching functionality is.



2.2 Infrastructure Services

a.The Google Play Store will inform the publisher/users of download counts, user ratings, and user feedback via the account user accessing the WaiNZ account within the Google Play Store's own facilities.

b. Crashlytics has been implemented to report application crashes via e-mail, and to display detailed information regarding the crash, i.e. (Phone used, API level, Code Line, Exception Message).

3. System Design

3.1 Documentation and Naming Standards

1. For the java code, documentation should follow JavaDoc standards, and naming should follow standard java conventions.
2. For the XML, W3C standards apply.
3. Comments should imply code function and purpose.

3.2 Software development tools

1. Eclipse with ADT plugin including android and google API's level 18
2. Crashlytics
3. Diagrams may be drawn in any preferred UML tool.

3.3 Decomposition Description

1. The following diagrams depict events deemed important to the flow of the application.
    a. App load
    b. When the user has selected an image from device's internal storage or a picture is taken using the device's camera.
    c. When the user selects an incident tag for their image
    d. When the user attempts to submit an incident report

fig3. Sequence Diagram outlining the events which occur when the app first starts. In summary the app sets up the UI, then checks if there are any cached images to be sent to the server for submission, as to which it then attempts to submit them.

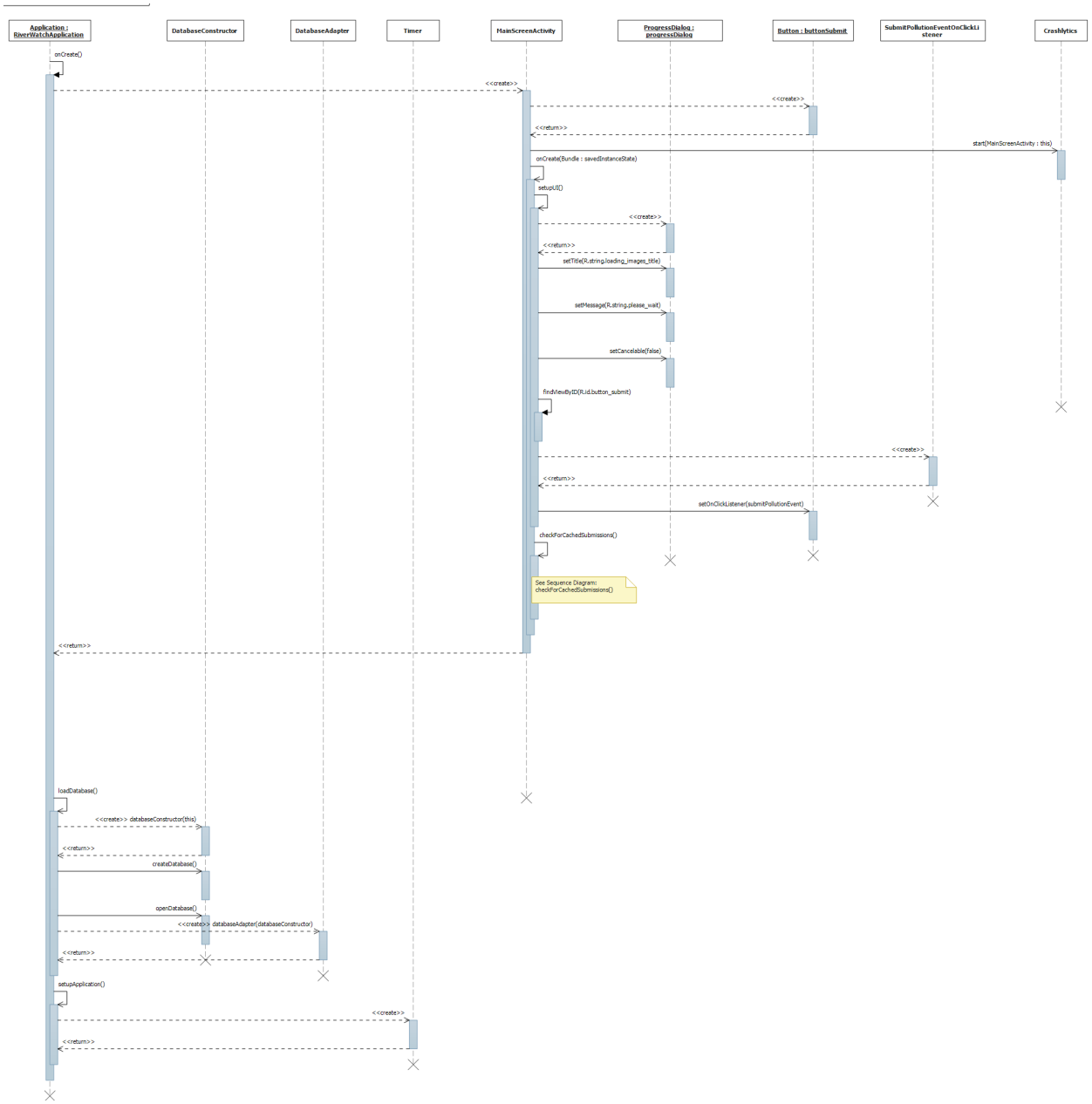


fig4. Sequence Diagram for the method checkForCacheSubmissions as referenced in *fig3.*

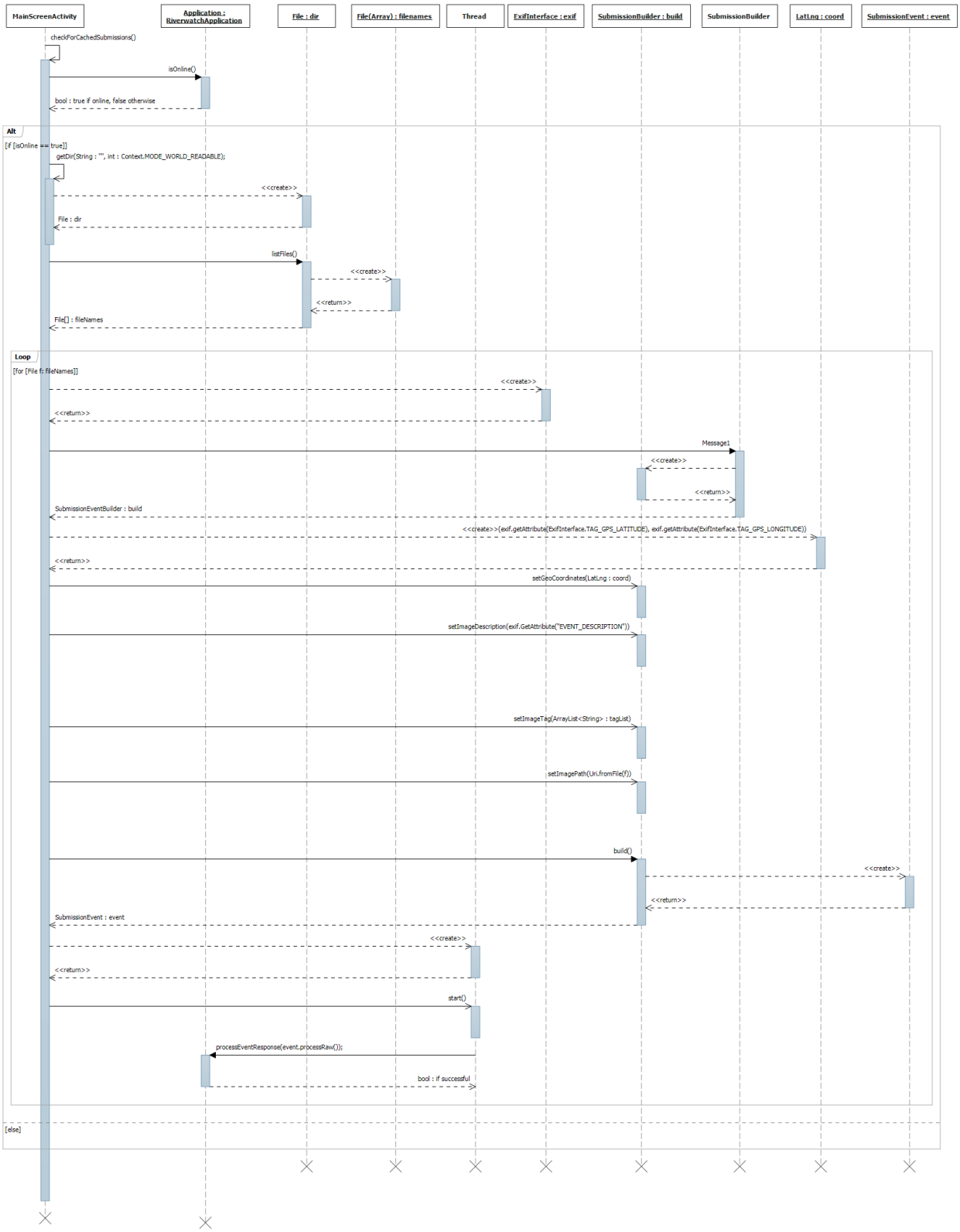


fig5. Sequence Diagram of Events when user selects an image either from the GalleryView or when an image is taken using the camera.

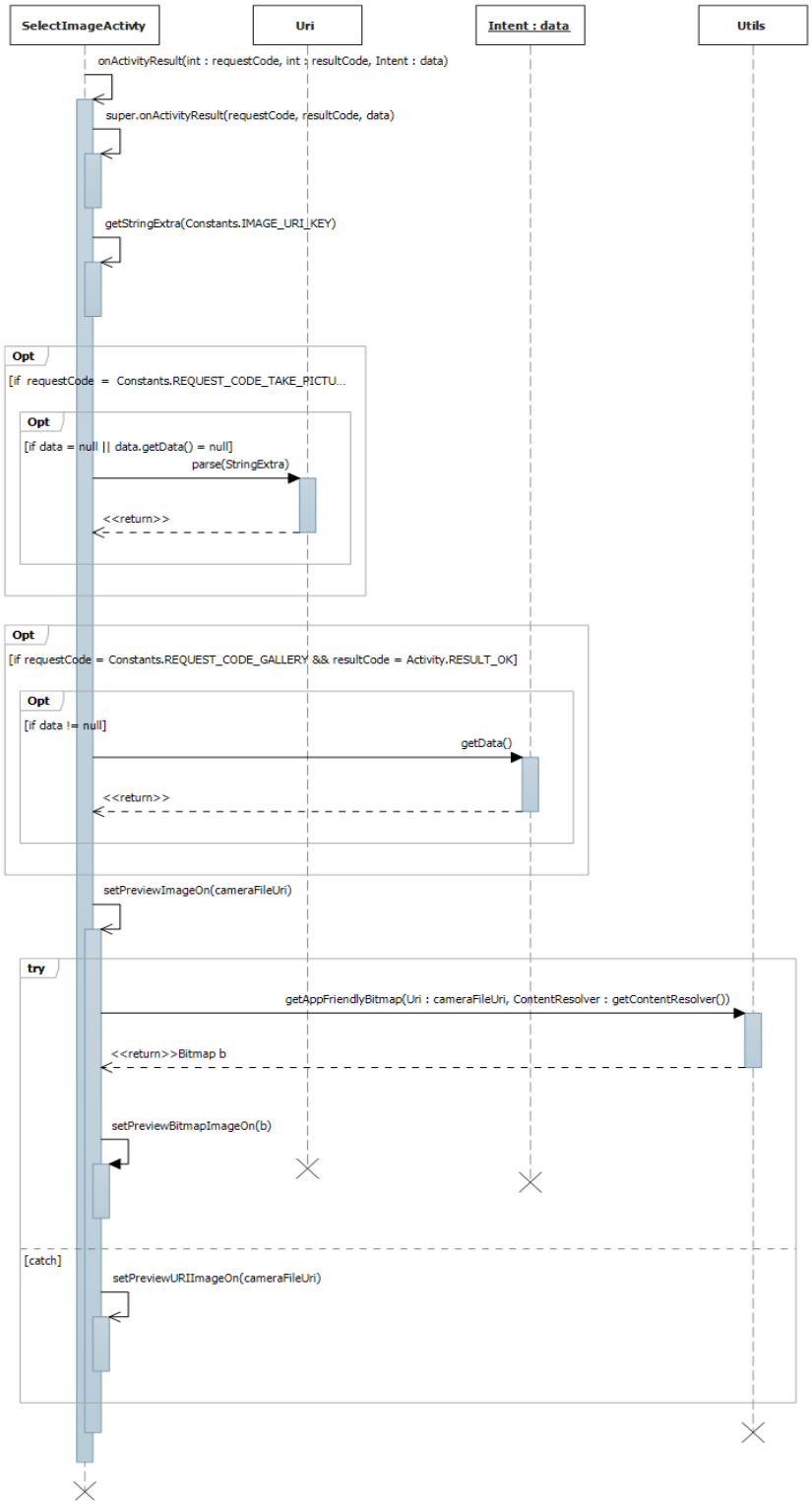
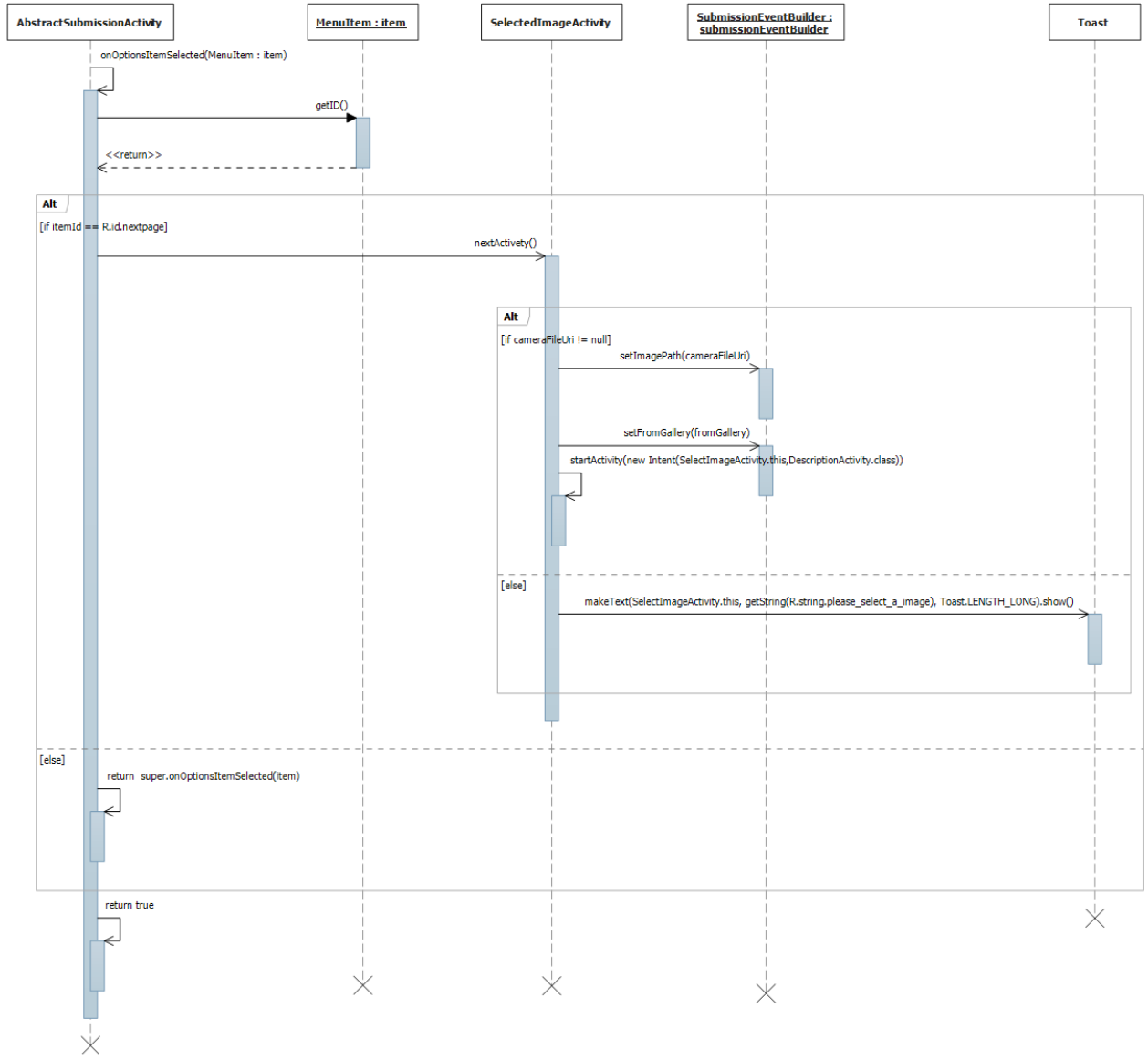


fig6. Sequence Diagram of Events when user selects an incident tag for the selected Image.



Document Signoff

Person	Signature	Date	Role
Tony Work			Author
Ben Vidulich			Author
Jason Pather			Author
Simon Clarke			Author
Hamish Cundy			Author
Grant Muir			Client
George Allan			Project Advisor
Winston Seah			Project Advisor
Lawrence Collingbourne			Project Advisor

Document Change Record

Date	Version	Author	Change Details