SWEN325 – Software Development for Mobile Platforms

SWEN325 – Assignment 1

# Contents

[Contents 2](#_Toc48918098)

[Introduction and Task Description 2](#_Toc48918099)

[Written Report (To Be Done Individually) (Worth 60%) 2](#_Toc48918100)

[What to submit. 2](#_Toc48918101)

[Application Architecture – TO DO (2 pages) 4](#_Toc48918102)

[Source code organisation 4](#_Toc48918103)

[UI Components 4](#_Toc48918104)

[Data 5](#_Toc48918105)

[Major External Components – TO DO (1 page) 6](#_Toc48918106)

[Map 6](#_Toc48918107)

[Camera 6](#_Toc48918108)

[Remote Server (using attached postgresql server) 6](#_Toc48918109)

[Framework Reflection (2 pages) – NOT COMPLETE 7](#_Toc48918110)

[Speed of iteration 7](#_Toc48918111)

[Framework Flexibility 7](#_Toc48918112)

[Access to native devices with plugins via Capacitor 7](#_Toc48918113)

[Deployment to device (Android) 7](#_Toc48918114)

[Usability Test Plan (3 pages) 8](#_Toc48918115)

[Application flow diagram 8](#_Toc48918116)

[Goals for the usability test – NOT COMPLETE 8](#_Toc48918117)

[Format of the usability test –NOT COMPLETE 8](#_Toc48918118)

[Target users (and how many) – NOT COMPLETE 9](#_Toc48918119)

[Tasks that users have to perform with the App (including screen shots) – NOT COMPLETE 9](#_Toc48918120)

[Appendix with screens 10](#_Toc48918121)

# Introduction and Task Description

## Written Report (To Be Done Individually) (Worth 60%)

The reports are assessed individually and therefore if you are in a group of two or three people, you will need to write it yourself and present a different usability test plan from your other group members.

### What to submit.

Submit a file called 325-a1-report-username.pdf that contains:

* Description of the overall architecture of your application. 2 Pages of A4, Worth 10% out of 60%
* Include a description of how you organised your source code to match your architecture.
* Description of how you utilised at least 1 major existing external component (e.g. Firebase but hopefully much more interesting one than that) to provide either persistent data storage or other major functionality with a description of how it was integrated into your App architecture. 1+ Page of A4, Worth 10% out of 60%
* Reflective report on Ionic as a framework for App design including its advantages and disadvantages. 2 Pages of A4, Worth 10% out of 60%
* Usability test plan 3+ Pages of A4, Worth 30% out of 60% that includes:
  + Goals for the usability test
  + Format of the usability test
  + Target users (and how many)
  + Tasks that users have to perform with the App (including screen shots)
* Finally, include an appendix with as many pages as there are separate screens in your app, with each page containing a screenshot of the screen and a one paragraph description of its design.

# Application Architecture

## Source code organisation

src

│ App.test.tsx

│ App.tsx

│ index.tsx

├───components

│ │ Default.css

│ │ ExploreContainer.css

│ │ ExploreContainer.tsx

│ ├───assetListItem

│ │ AssetListItem.tsx

│ ├───currentSession

│ │ CurrentSessionContainer.tsx

│ ├───home

│ │ HomeContainer.tsx

│ │ IntroductionContainer.tsx

│ ├───map

│ │ map.css

│ │ Map.jsx

│ ├───menu

│ │ Menu.css

│ │ Menu.tsx

│ ├───newAsset

│ │ AssetIdPhoto.tsx

│ │ NewAssetComponent.tsx

│ │ NewAssetContainer.tsx

│ │ SerialNumberPhoto.tsx

│ ├───previousSessions

│ │ PreviousSessionsContainer.tsx

│ ├───settings

│ │ SettingsContainer.tsx

│ └───upload

│ UploadContainer.tsx

│ UploadInformationContainer.tsx

├───contexts

│ AssetsContext.tsx

│ SettingsContext.tsx

├───data

│ │ index.tsx

│ ├───cloud

│ │ Cloud.tsx

│ └───local

│ Local.tsx

├───pages

│ Page.css

│ Page.tsx

├───secrets

│ googleMaps.js

├───theme

│ variables.css

└───types

Asset.tsx

index.tsx

Settings.tsx

### UI Components

The general application architecture splits the application into 6 major sections, four on the device and two located in the cloud. The code on the phone is separated into UI and business/data.

The UI is built from the app holding a single Page element, and within that page element containers are turned on and off as the user navigates the UI.

All of the containers sit within the ‘Components’ folder within their own named folder, this helps to keep the files organised in what might otherwise be a large list of files without context. Some of the elements are reused and so have been brought to the top level e.g. assetListItem, which is used in both the “Current Session” and “Previous Session” containers.

Other elements are only used in a single component and so are kept in the same folder, such as the SerialNumberPhoto and AssetIdPhoto elements. This helps to avoid a cluttered folder structure where all components have their own folders. Creating a separate element is only done if it is significantly complex – so things like Toasts are kept within the main component.

### Data

#### Types

Types are the interfaces used by objects in the application. These have all been centralised in the types folder and are accessed by importing that folder and the associated type. Types in the application are Asset, Assets and Settings.

#### Context

Context is used to ensure consistency across the application without passing props up and down through components, these contexts are kept in their own folder and reference both the Types and the Data Locations.

#### Data Locations

Data locations are abstractions for access to local and cloud data. In the case of the Cloud data location, it allows for the objects retrieved to be converted to the local representations. There are specific methods in the data locations for dealing with both Asset and Settings.

#### Secrets

Secrets are a special type of data that includes API keys and data that needs to be kept from replication. In an ideal situation this would be kept as part of the environment variables and so not included with source code.

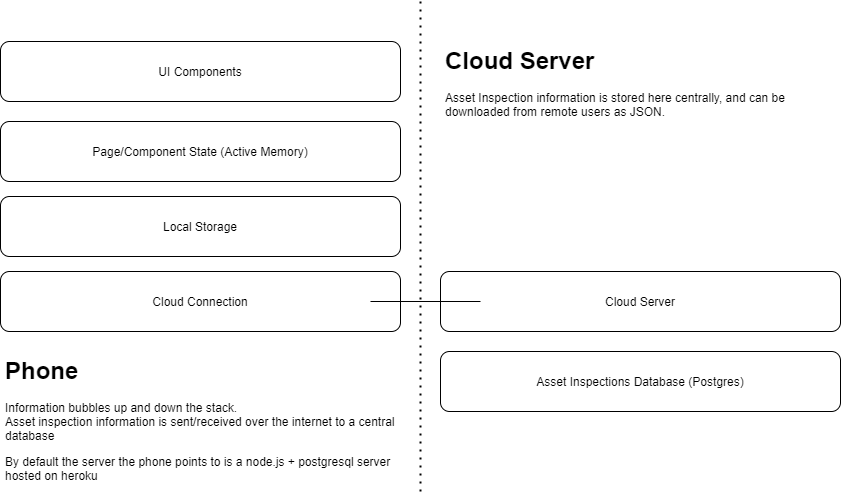


Figure 1: Data flow within the application

# Major External Components – TO DO (1 page)

For External Component, you **need to** describe how your way of implementing it fits into your proposed architecture and provide technical details of how it works and **justify** your choices and why it is the best way to fit the component into your app architecture and design.

Description of how you utilised at least 1 major existing external component (e.g. Firebase but hopefully much more interesting one than that) to provide either persistent data storage or other major functionality with a description of how it was integrated into your App architecture. 1+ Page of A4, Worth 10% out of 60%

## Map

The map is a useful visualisation of the GPS coordinates which are pulled from the device. Without this visualisation identifying a problem would be difficult and this improves the experience as the user inputs data.

## Camera

One of the major benefits of using a phone to take this type of information is that a camera can be used to capture information and make manual processing easier later on.

The camera is only used within one of the screens (New Asset), it includes taking the photo and allowing the user to review the photo when inputting the manual data later on.

## Remote Server (Cloud) (using attached postgresql server)

The remote server is a node.js server with an attached postgresql server attached, allowing for asset inspections to be saved and reviewed later. As the aim for this application is to act as a handset for gathering information the central and remote server is a very important part of the architecture.

In many cases companies performing asset inspections will have existing lists of assets, so being able to export the asset information out in an easy to use format is important. This is done with a JSON to response to GET requests to the /assets URL.

Remote Server is a Data Location and has had all information about working with the remote server centralised to a single file within the Data Locations folder.

# Framework Reflection (2 pages) – NOT COMPLETE

For Framework Reflection, you are expected to provide a lot of technical details, examples, and justifications for how the Ionic or React Native helped or hindered your development. It is essential to not just describe how it works and whether you liked it but rather justify and show technical examples of how it affected your application design and what further improvements or insight you could have hypothetically provided to framework developers.

Reflective report on Ionic as a framework for App design including its advantages and disadvantages. 2 Pages of A4, Worth 10% out of 60%

## Speed of iteration

Iteration speed was incredibly fast with the use of the Ionic framework, with hot reload on save using the `ionic serve` command, the speed that changes could be tested was very quick.

## Framework Flexibility

The option to use React and Angular allowed for me to leverage my existing skills with React, making components that could be easily included and modified. With the new React hooks this became even easier than dealing with props and modifying state at every level.

## Access to native devices with plugins via Capacitor

Using the plugins such as {Camera} made working with the camera relatively easy, instead of trying to manage the webpage plugins needed for cameras and videos a call was made to the plugin and the result was quite easy to use.

## Deployment to device (Android)

This application is designed to be used on a mobile phone, being able to test and deploy to a mobile device is crucial. Using the commands

ionic build

ionic capacitor run android external

I was able to build and then deploy straight onto an attached android device to test my application.

The application is installed on the attached device and allows for long term testing without having to go through any app store. Because the app is installed the device doesn’t need to be connected to the deploying PC, it can be moved to test all of the situations one might expect the device to be in and will be running on real hardware.

## Starter Projects

Starter projects for Ionic made starting a basic app really fast, being able to keep and modify parts quickly. Adding navigation into a website can be difficult to do in a way that is beautiful and seamless.

## Components

Components like toasts were easy to add in and treat like an HTML element, so much so that it was pointless to try to turn them into free standing components associated with the container.

# Usability Test Plan (3 pages)

## Application flow diagram

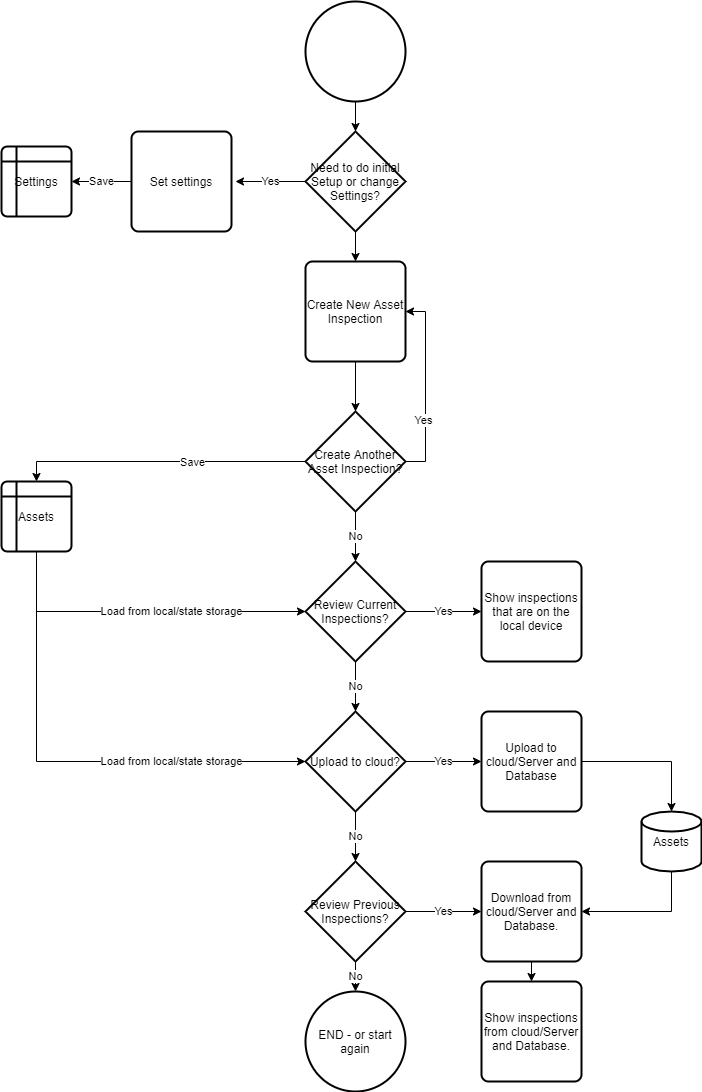


Figure 2: Application flow diagram, showing choices and flow through the application

## Goals for the usability test – NOT COMPLETE

The goal of the usability test is to identify design flaws, to allow for them either to be corrected in their entirety or to be mitigated with other elements.

## Format of the usability test –NOT COMPLETE

The user will need to add three assets to the remote database and confirm that they have been added.

Feedback will be qualitative for both positive and negative aspects of each of the pages and tasks.

Broken into multiple tasks

* Setup
* Create new asset inspections
* review and clean asset inspections
* upload to cloud
* review previous inspections

Each task will have a comments section.

By default, testers will be asked to perform the tasks without any assistance, but will be able to ask for the instructions (tasks that users have to perform with the app – including screenshots) if they want to.

An overall section will be included.

## Target users (and how many) – NOT COMPLETE

The target users are people who have been tasked with performing asset inspections.

The primary group this is aimed at is technical professionals maintaining asset records, but could also be used by any others that need a record of assets with both time, place and some important details.

2 users will be asked to perform the test

## Tasks that users have to perform with the App (including screen shots)

### Setup

1. Open the app

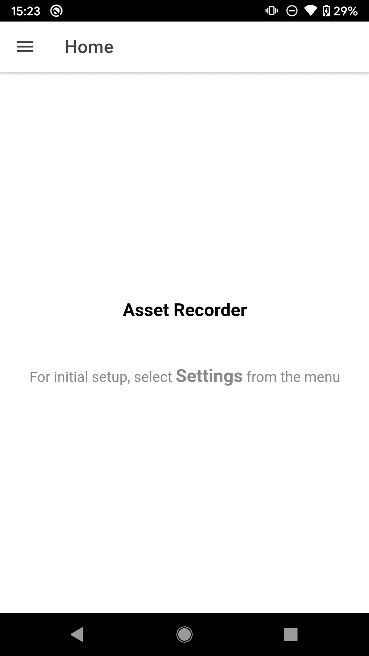
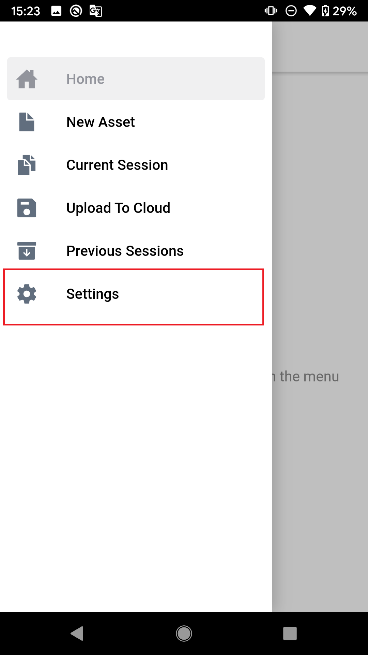


Figure : Home Screen providing information about initial setup

1. Navigate to settings



1. Check or modify settings as needed

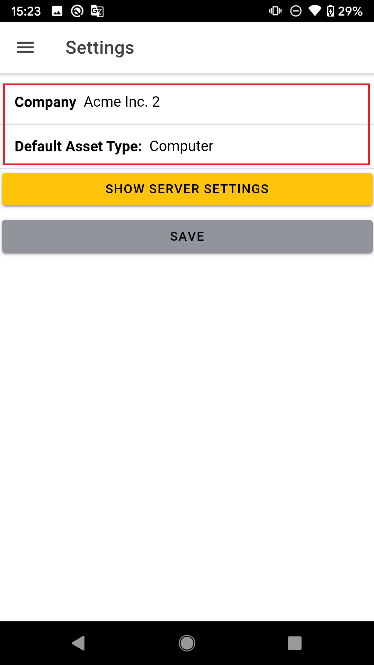


Figure : Application settings for the asset inspections that will be created

1. If an alternative server has been set up, or if all of the elements on the server need to be removed, change the server settings

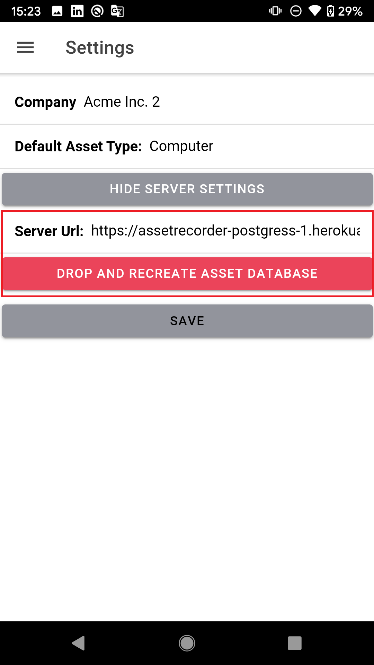


Figure : Server URL and Drop and Recreate Database button

### Create new asset inspections

1. Navigate to New Asset through the menu/hamburger button

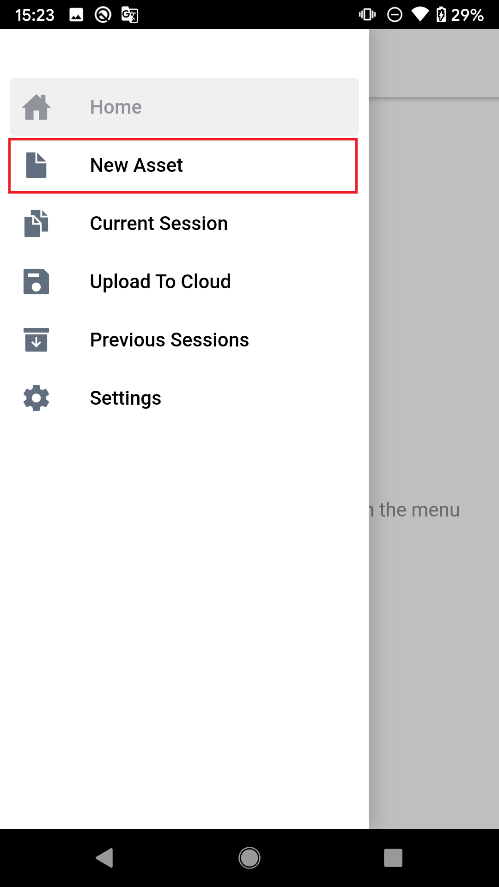


Figure : Navigation to New Asset

1. Fill out details including updating the location.

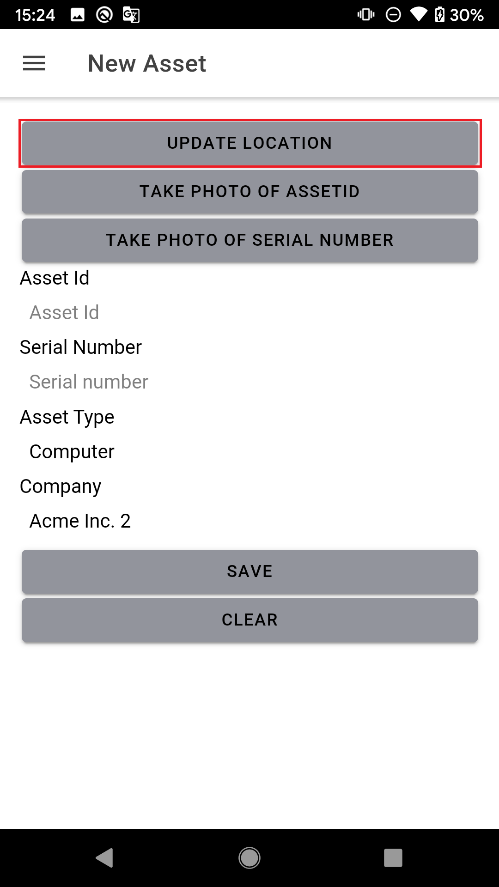


Figure : Update location for New Asset

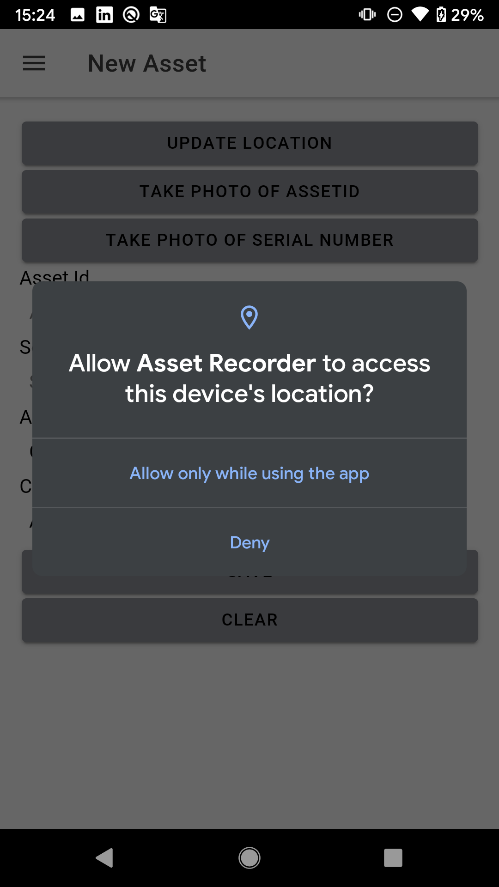


Figure : First time alert for using the device location

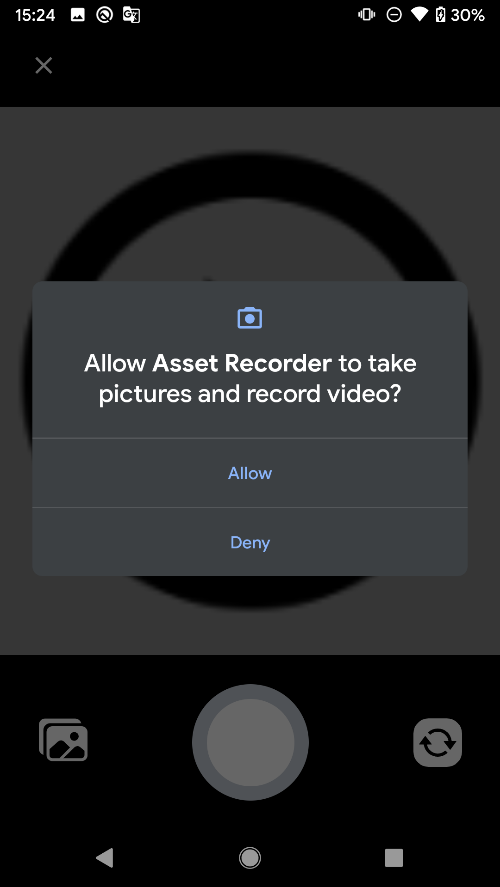


Figure :First time alert for using the device Camera

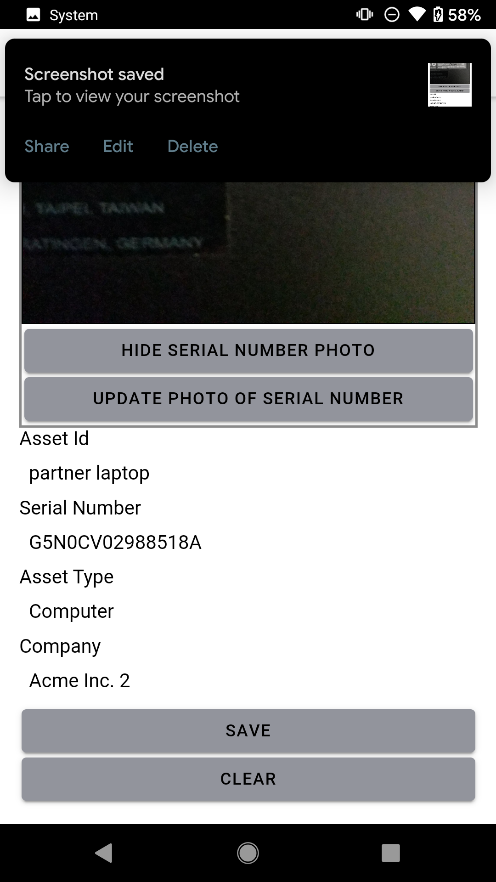


Figure : Completed asset inspection information

1. Save the asset

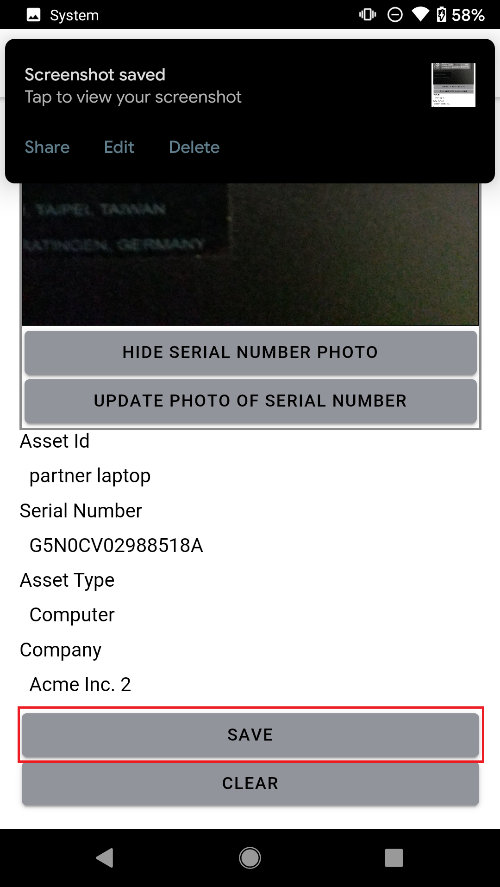


Figure : Save button for asset inspection

1. Continue to save and record new assets as needed for this current session

### Review and clean current asset inspections

1. Navigate to current session

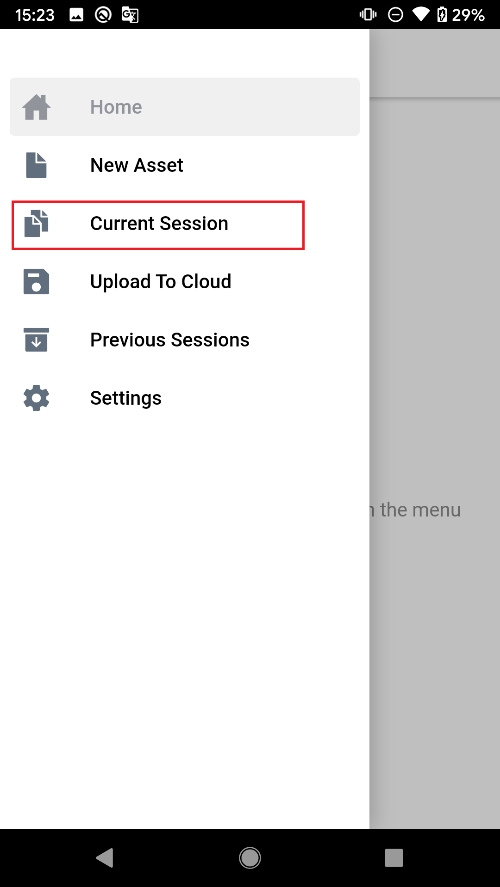


Figure : Navigation to Current Session

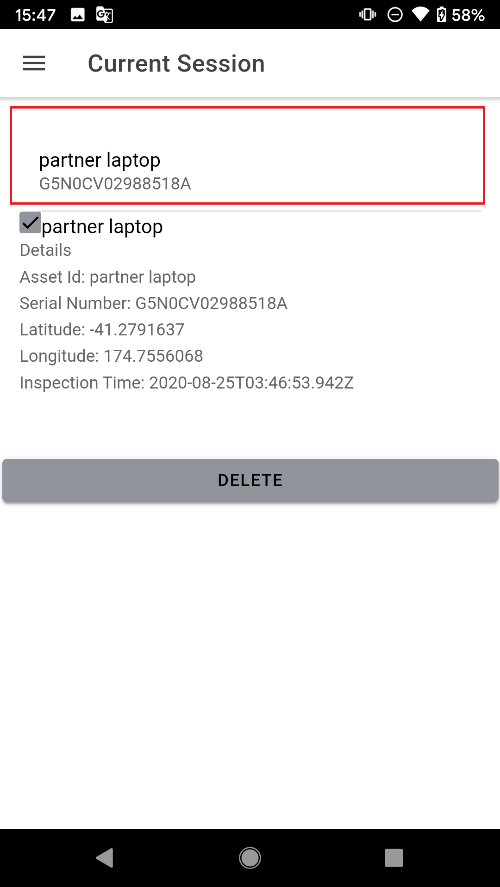
1. Individual items can be clicked on to view more information if needed

Figure : Showing the current session and both an expanded (and selected) asset inspection, and a highlighted minimal overview of an asset inspection

1. Put a check in the checkbox for any item to be deleted
2. Press delete to delete the items checked

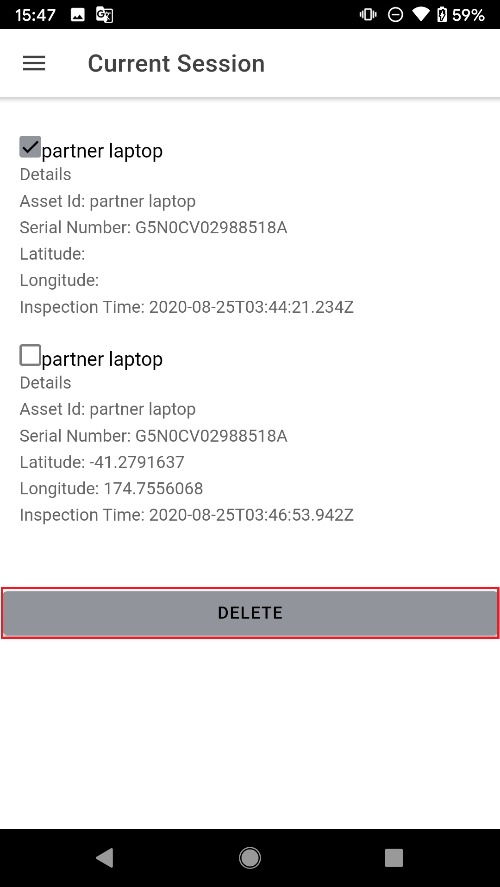


Figure : Duplicate with less information found, click delete to remove from the Current Session

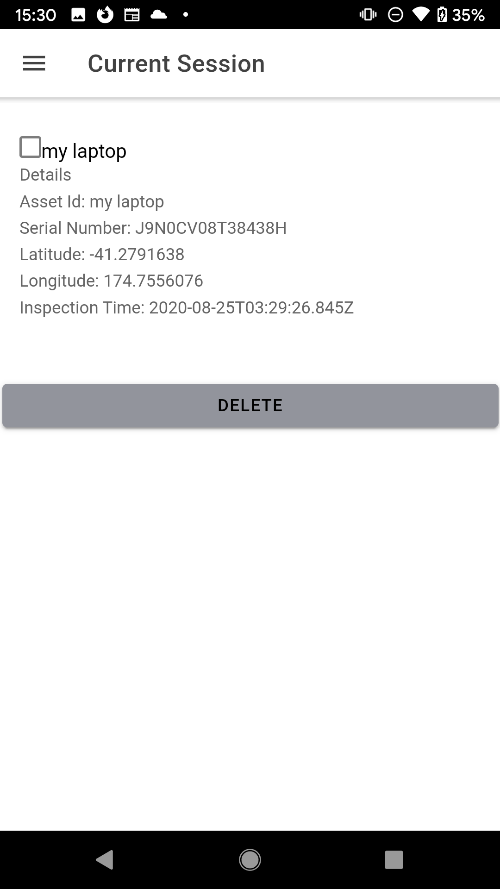


Figure : Current Session with information

### Upload current session asset inspections to cloud

1. Navigate to upload to cloud

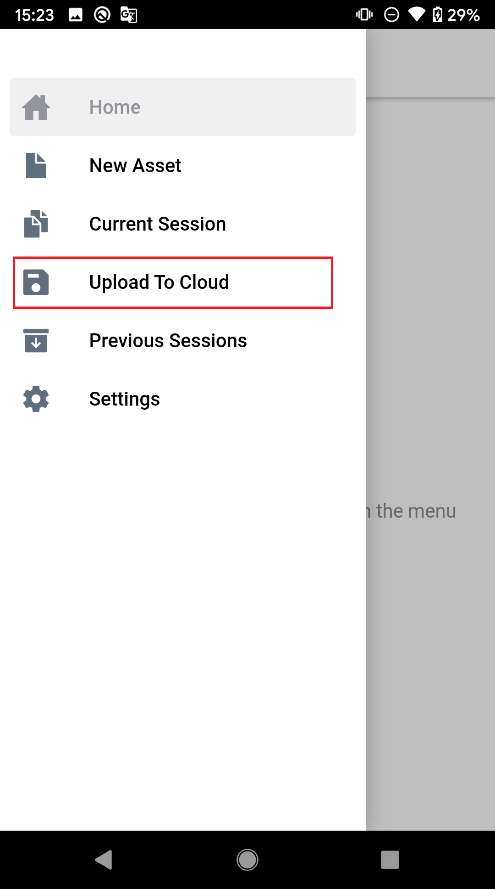


Figure : Navigation to Upload to Cloud

1. Click Upload to cloud to upload to cloud and clear asset inspections from the local device

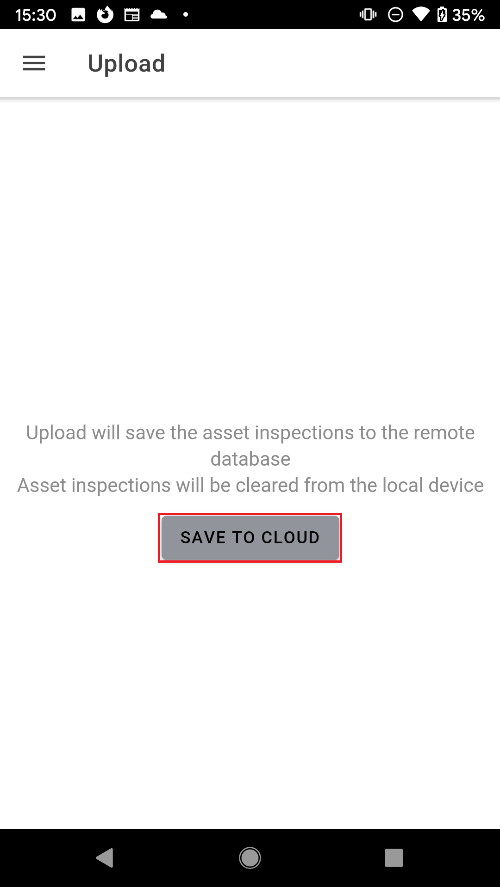


Figure : Save To Cloud button to save to cloud and clear asset inspections from the local device

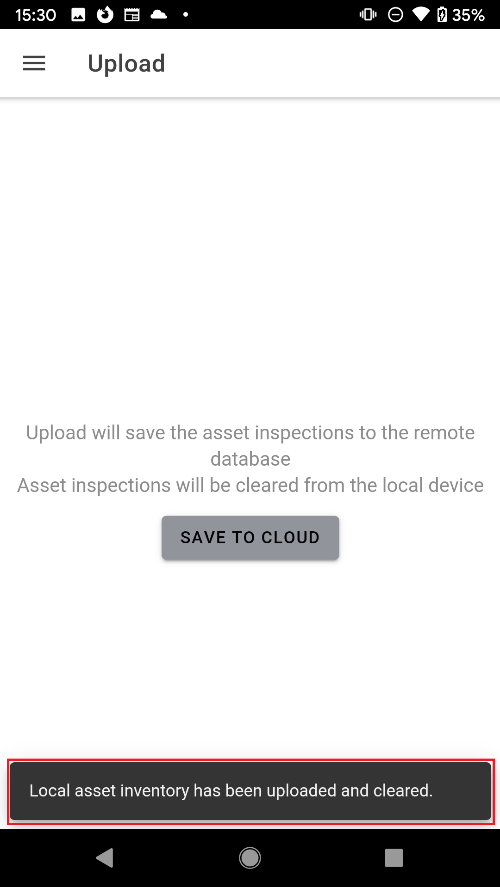


Figure : Toast to notify user that the upload was completed successfully

### (Optional) Review previous sessions

1. Navigate to previous sessions

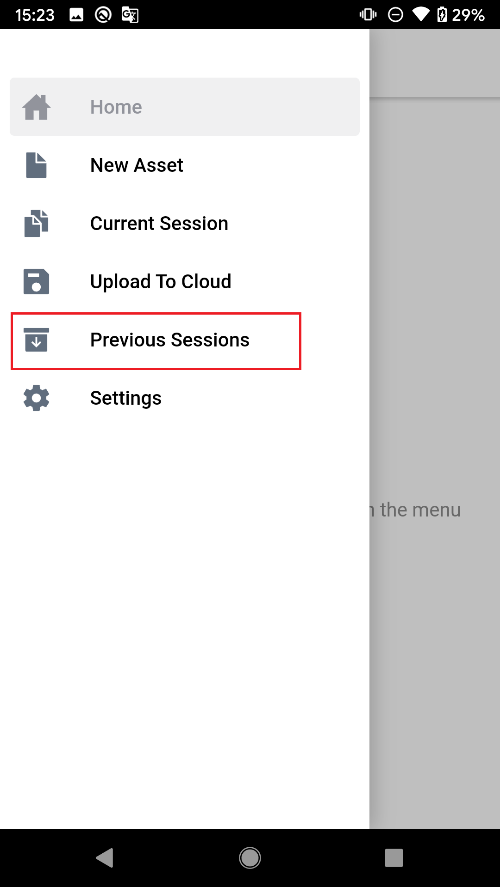


Figure : Navigate to Previous Sessions

1. Click on any asset inspection to see more details

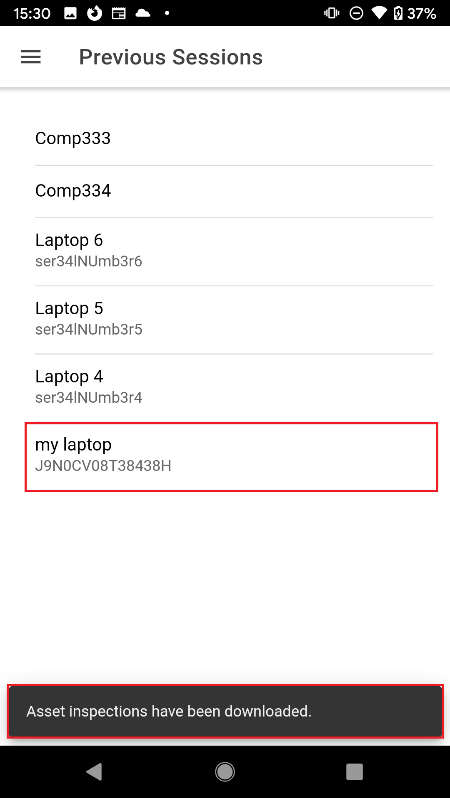


Figure : Toast alerting the user that the Asset Inspections have been downloaded and a highlighted element that can be opened to review more details

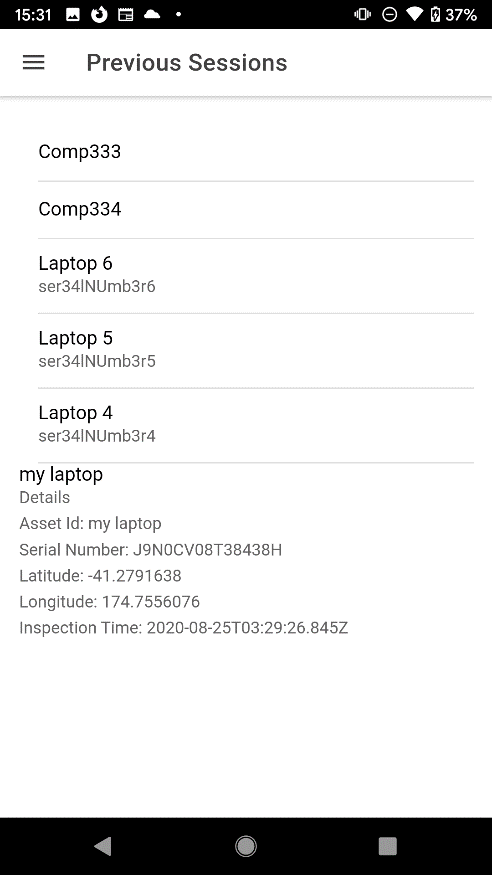


Figure : Expanded Asset Inspection

# Appendix with screens

Finally, include an appendix with as many pages as there are separate screens in your app, with each page containing a screenshot of the screen and a one paragraph description of its design.