SWEN325 – SOFTWARE DEVELOPMENT FOR MOBILE PLATFORMS

SWEN325 – Assignment 1

Gene Culling – 300476456

Contents

Contents 2
Introduction and Task Description 4
    Written Report (To Be Done Individually) (Worth 60%) 4
        What to submit. 4
Application Architecture 5
    Source code organisation 5
        UI Components 5
        Data 6
Major External Components 7
    Map 7
    Camera 7
    Storage - local storage (Local) 7
    Storage - Remote Server (Cloud) (using attached postgresql server) 7
    Justifications 7
Framework Reflection (2 pages) 8
    Framework Flexibility 8
    Starter Projects 8
    Components 8
    Speed of iteration 8
    Access to native devices with plugins via Capacitor 8
    Deployment to device (Android) 8
    Ionic Speed 9
    Comparison of Ionic to React Native 9
    Potential insights/improvements for future framework developers 9
Usability Test Plan (3 pages) 10
    Application flow diagram 10
    Goals for the usability test 10
    Format of the usability test 11
    Target users (and how many) 11
        Prototype phase 11
        Alpha phase 11
        Beta phase 11
        Release phase 11
    Tasks that users have to perform with the App (including screen shots) 12

Setup 12
Create new asset inspections 14
Review and clean current asset inspections 17
Upload current session asset inspections to cloud 19
(Optional) Review previous sessions 20
Appendix with screens 22
Home 22
New Asset 23
Current Session 24
Upload to Cloud 25
Previous Sessions 26
Settings 27
Menu 28

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.

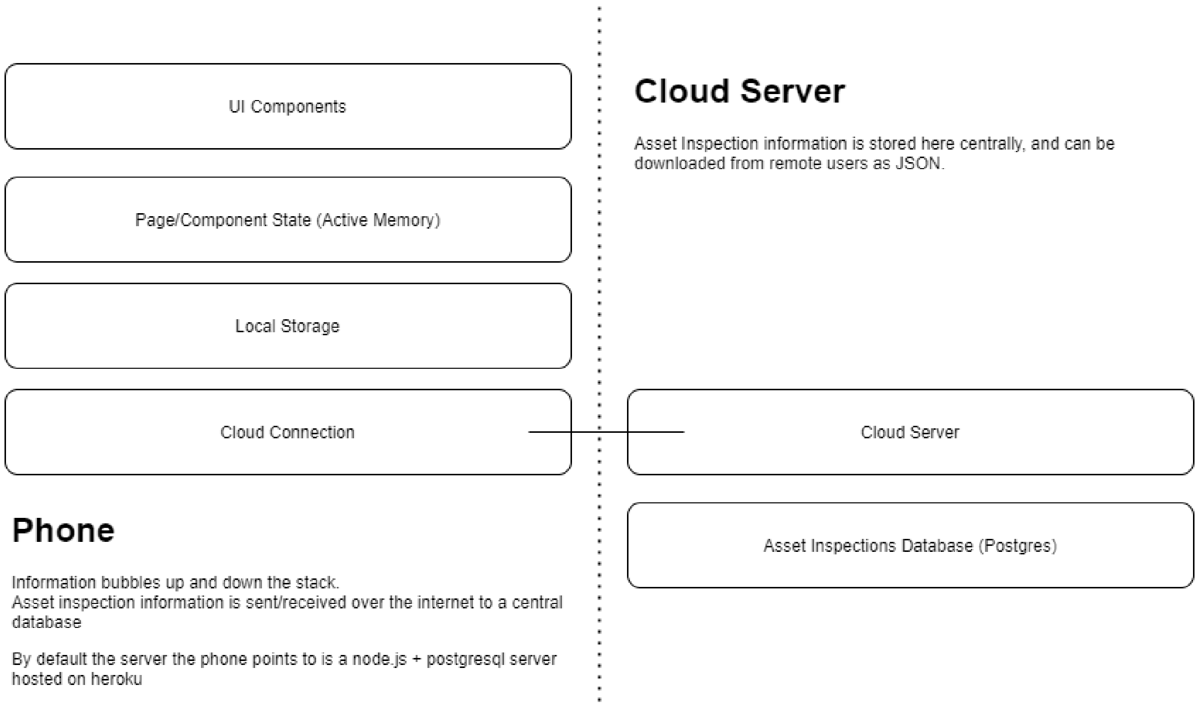

UI Components
Page/Component State (Active Memory)
Local Storage
Cloud Connection
Phone
Information bubbles up and down the stack.
Asset inspection information is sent/received over the internet to a central database
By default the server the phone points to is a node.js + postgresql server hosted on heroku
Cloud Server
Asset Inspection information is stored here centrally, and can be downloaded from remote users as JSON.
Cloud Server
Asset Inspections Database (Postgres)

Figure 1: Data flow within the application

Major External Components

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.

Storage - local storage (Local)

Local storage provides a way to keep information on the device even after reboots/unexpected shutdowns. The local storage component is the place that assets are saved to initially and from which are uploaded to the cloud server as part of the "upload to cloud" task.

Storage - 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.

Justifications

Map and camera are only used as part of the New Asset inspection task, and so are kept in the folder with the New Asset container. Camera is a plugin that is used by both the Asset Id Photo and Serial Number Photo components and is imported separately for both – in the future there is an aim to potentially enable barcode scanning or OCR which may need to be addressed differently for each of the photo components.

The Remote Server(Cloud), is a "Data Location" and is kept in it's own folder, accessible from the data locations import in a way very similar to the Local data location. The local storage is also a "Data Location" and is accessed in much the same way as the Cloud server.

Framework Reflection

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.

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.

For my needs I chose to use the Menu setup, the slide out bar allowed for a lot of additional pages and navigation by the menu was relatively simple.

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.

Speed of iteration

Iteration speed was incredibly fast with the use of the Ionic framework, with hot reload on save using the command

```
ionic serve
```

changes could be tested very quickly.

Access to native devices with plugins via Capacitor

Using the plugins such as Camera and Storage 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.

It may have been easier to work with local storage for the web browser directly instead of through the Storage component.

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.

Ionic Speed

As Ionic acts as a locally served webpage there are potential issues with performance in the case of games or elements that rely on fast feedback between the app and the underlying hardware. With my app these are not important considerations and may not have been an issue even on a remote webpage.

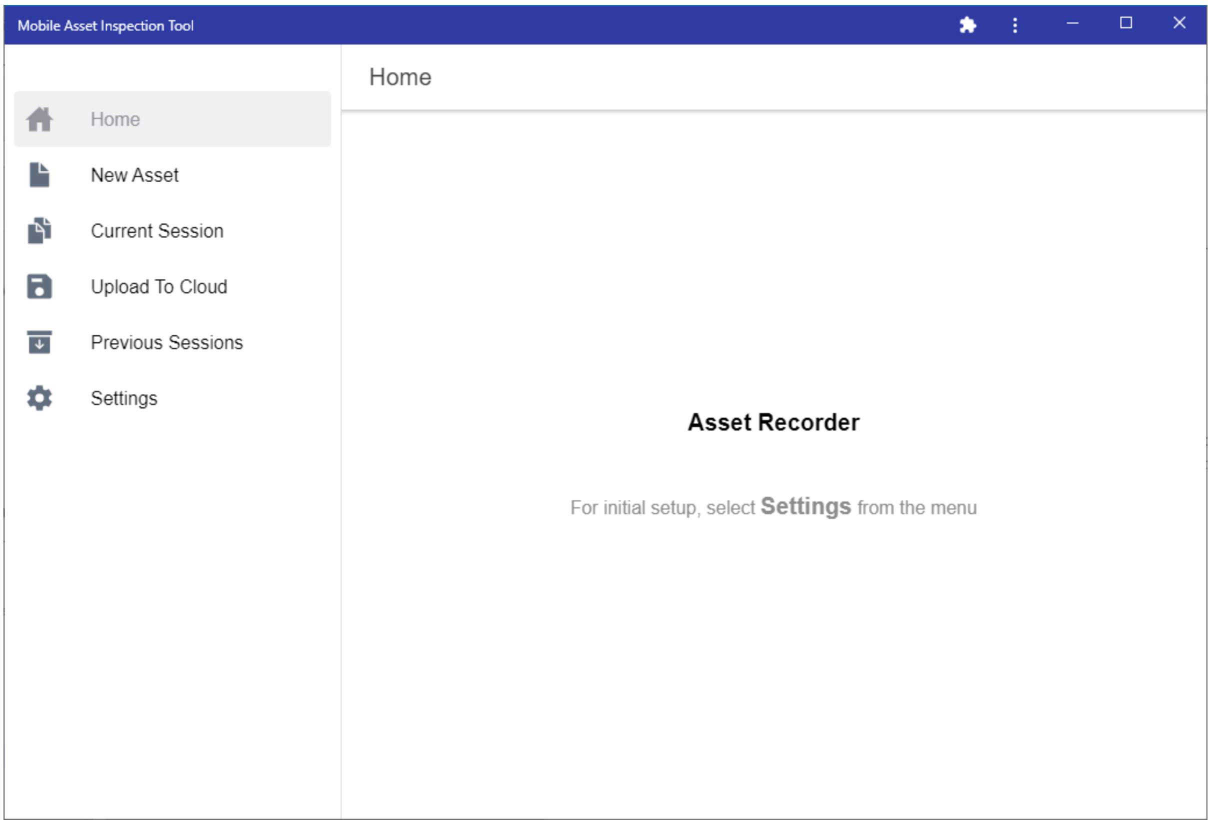
Comparison of Ionic to React Native

I have not yet used react native and so can't compare the two.

Potential insights/improvements for future framework developers

A lot of the work has already been done to make this work as a Progressive Web App, including the build option adding elements in to make it work as a PWA, with a little more extension the build process could create a full Progressive Web app that is installed/installable with certificate and web server to be deployed.

Using an existing server, this app has been deployed to my local PC and is able to run offline. This was deployed from the website and gives an easy way to let users access the application and use it in the future.



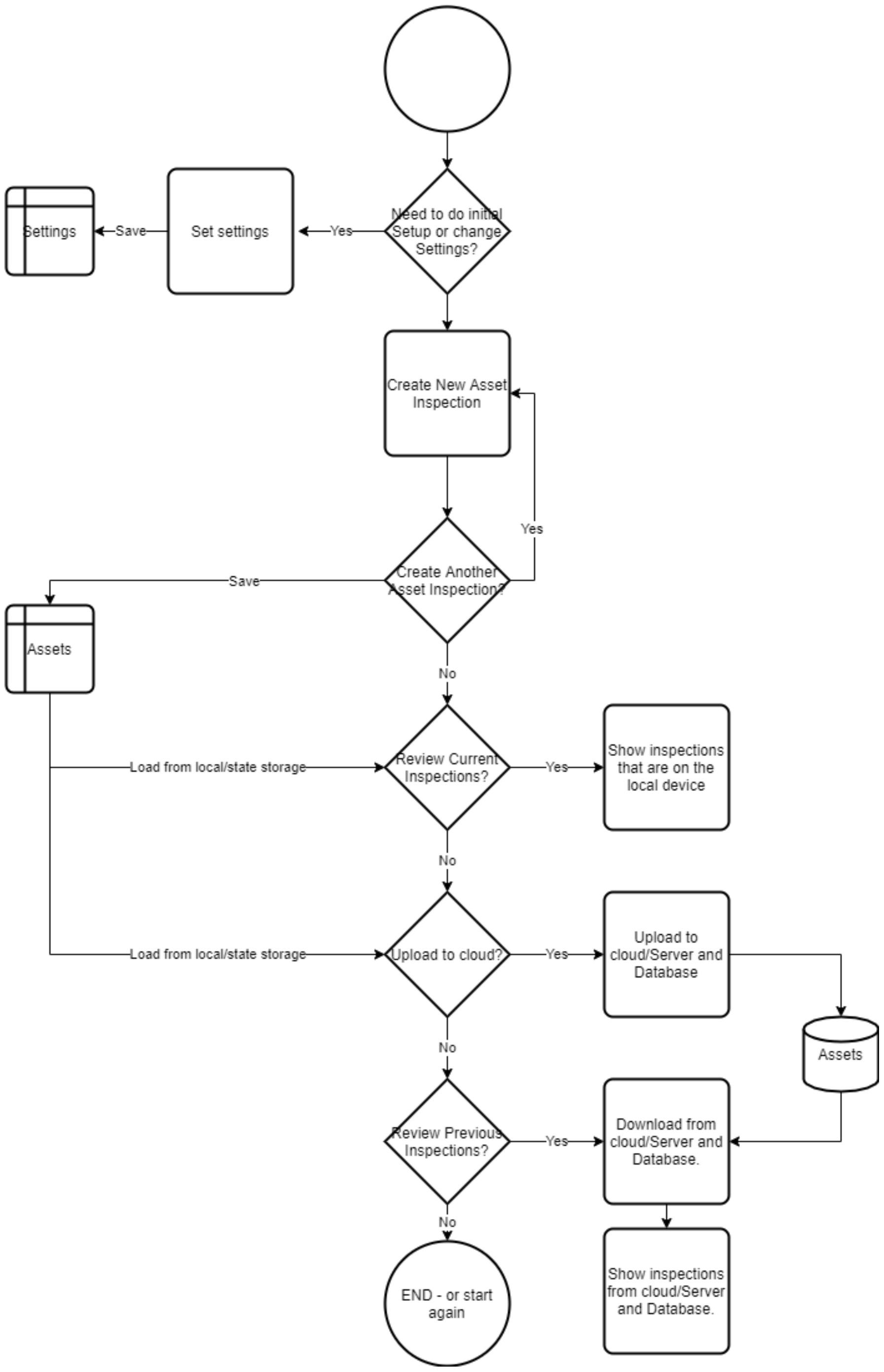
Asset Recorder

For initial setup, select **Settings** from the menu

Figure 2: Progressive Web App for the Asset Recorder application

Usability Test Plan

Application flow diagram



Need to do initial Setup or change Settings?
Yes
Set settings
Save
Settings
Create New Asset Inspection
Create Another Asset Inspection?
Yes
Save
Assets
No
Review Current Inspections?
Load from local/state storage
Yes
Show inspections that are on the local device
No
Upload to cloud?
Load from local/state storage
Yes
Upload to cloud/Server and Database
Assets
No
Review Previous Inspections?
Yes
Download from cloud/Server and Database.
Show inspections from cloud/Server and Database.
No
END - or start again

Figure 3: Application flow diagram, showing choices and flow through the application

Goals for the usability test

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

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 and an overall section will be included.

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.

This format is used for all prototype and alpha phases.

Target users (and how many)

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

Target users will already know what Assets, Asset Ids and Serial numbers are.

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.

Prototype phase

Three users will be asked to perform the test and fill out the associated documents in the prototype phase. While significant flaws are found in the prototype phase small and fast usability tests will be performed with small numbers of new users, with the feedback being incorporated into the next iteration of the prototype.

Alpha phase

Once completing a testing phase with no issues from new users a larger group of 30 technical users will be tested.

Two IT managers will also be asked for qualitative feedback on both UI and the tasks expected to be performed. Changes will be made according to feedback.

Beta phase

Released to a small company or a subsection of a large company with a subsidised price, when released into the company there is assumed to be flaws or features that will need to be identified in real world situations, this could include remote locations, old hardware, expensive network connections impacting the use of the app, more information needing to be captured by the application etc.

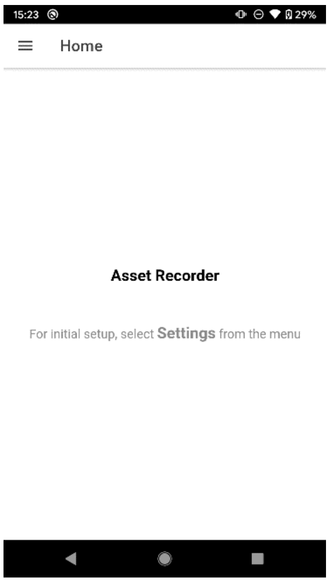
Release phase

Release phase assumes that most of the large-scale bugs have been found and that there may still need to be some changes to usability for some users or situations.

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

Setup

1. Open the app



Asset Recorder

For initial setup, select **Settings** from the menu

Figure 4: Home Screen providing information about initial setup

2. Navigate to settings

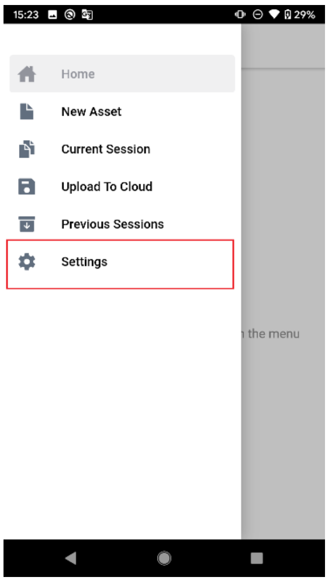
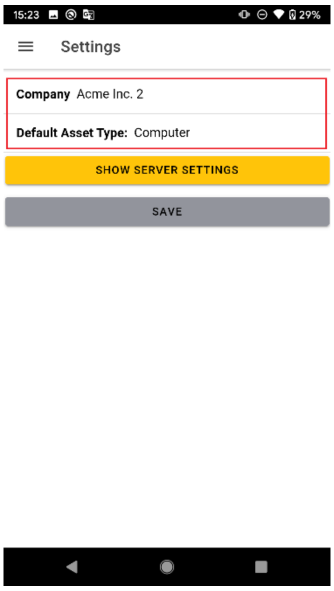


Figure 5: Settings menu object highlighted

3. Check or modify settings as needed


15:23 29%

Company Acme Inc. 2

Default Asset Type: Computer

Figure 6: Application settings for the asset inspections that will be created

Create new asset inspections

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

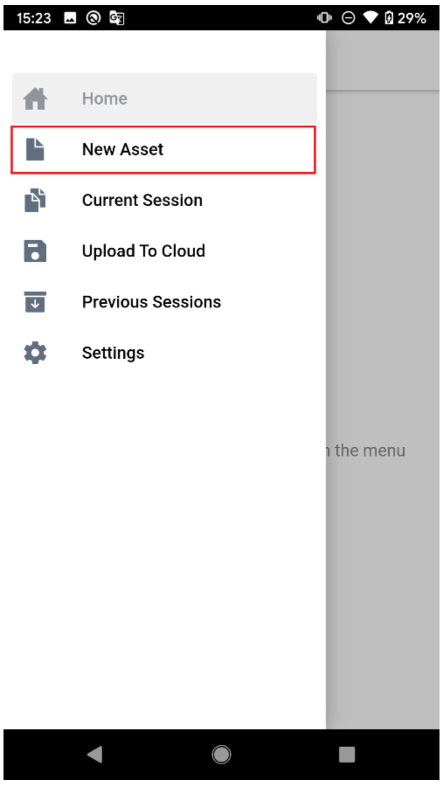
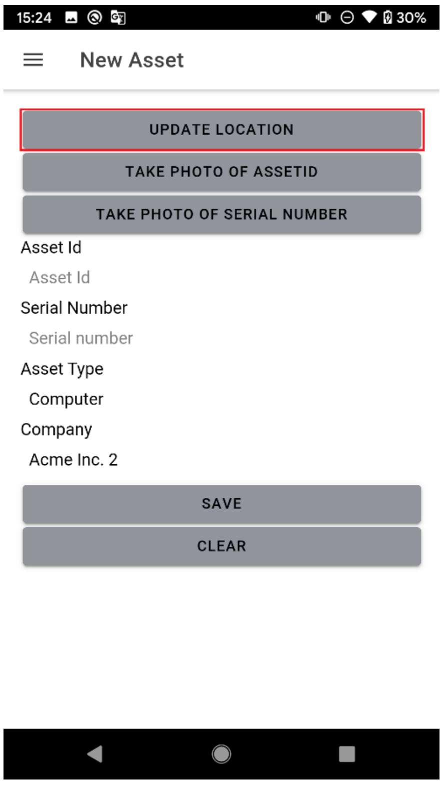


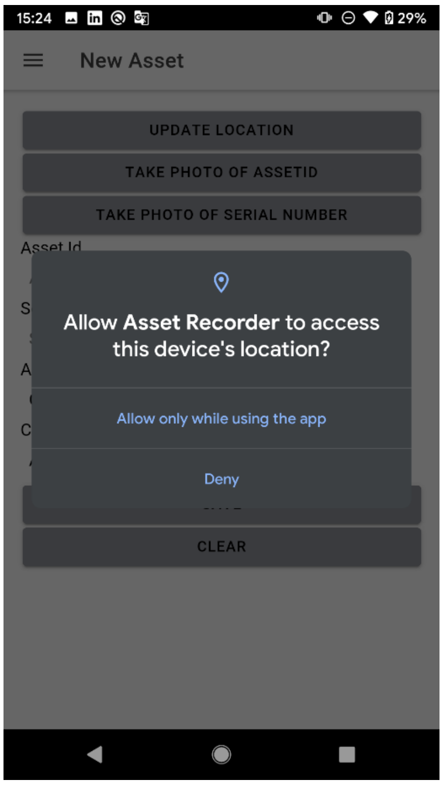
Figure 7: Navigation to New Asset

2. Fill out details including updating the location.



Asset Id
Asset Id
Serial Number
Serial number
Asset Type
Computer
Company
Acme Inc. 2

Figure 8: Update location for New Asset


15:24 29%

New Asset

UPDATE LOCATION

TAKE PHOTO OF ASSETID

TAKE PHOTO OF SERIAL NUMBER

Asset Id

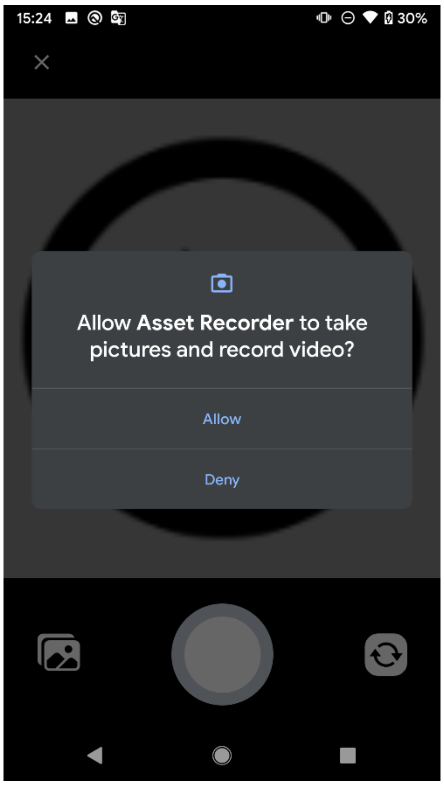
Allow Asset Recorder to access this device's location?

Allow only while using the app

Deny

CLEAR

Figure 9: First time alert for using the device location

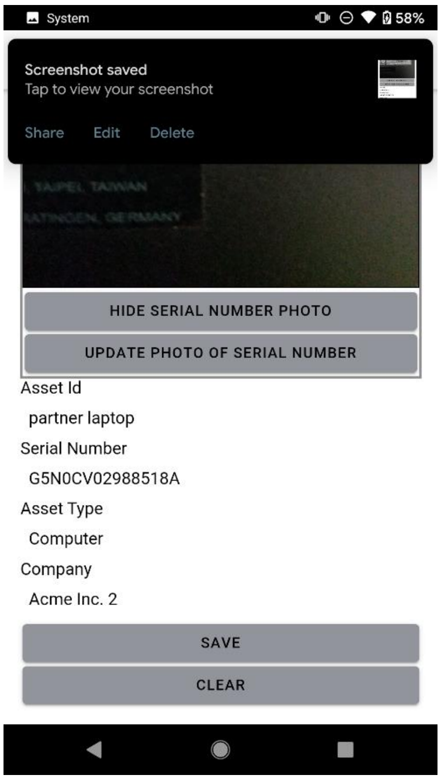

15:24 30%

Allow Asset Recorder to take pictures and record video?

Allow

Deny

Figure 10:First time alert for using the device Camera


System 58%

Screenshot saved
Tap to view your screenshot

Share Edit Delete

HIDE SERIAL NUMBER PHOTO

UPDATE PHOTO OF SERIAL NUMBER

Asset Id

partner laptop

Serial Number

G5N0CV02988518A

Asset Type

Computer

Company

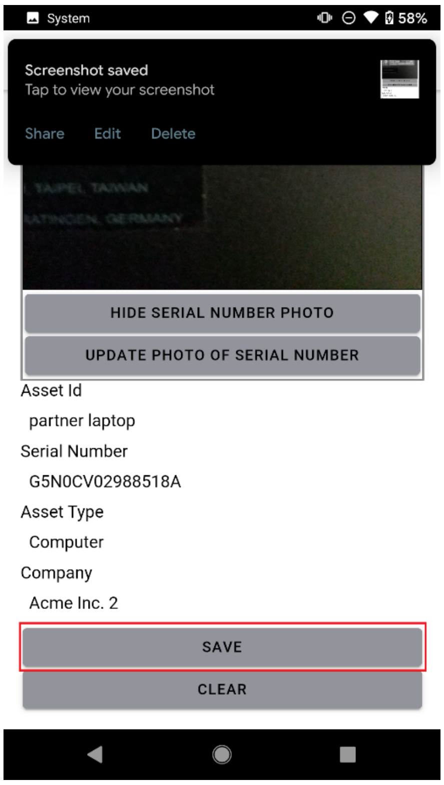
Acme Inc. 2

SAVE

CLEAR

Figure 11: Completed asset inspection information

3. Save the asset


System 58%

Screenshot saved
Tap to view your screenshot

Share Edit Delete

HIDE SERIAL NUMBER PHOTO

UPDATE PHOTO OF SERIAL NUMBER

Asset Id

partner laptop

Serial Number

G5N0CV02988518A

Asset Type

Computer

Company

Acme Inc. 2

SAVE

CLEAR

Figure 12: Save button for asset inspection

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

Review and clean current asset inspections

1. Navigate to current session

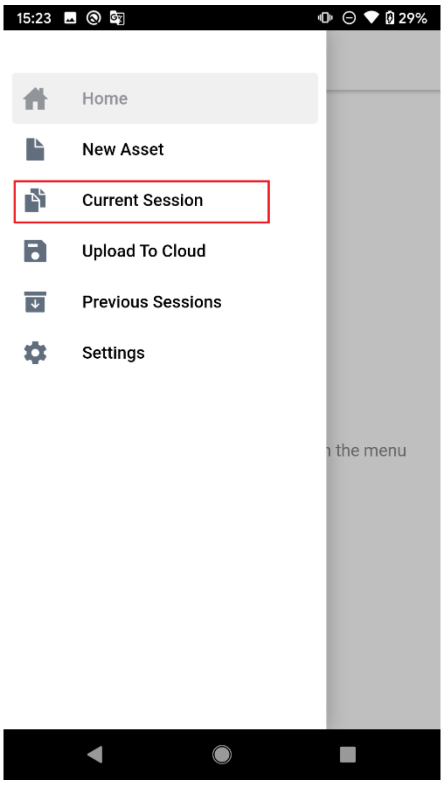
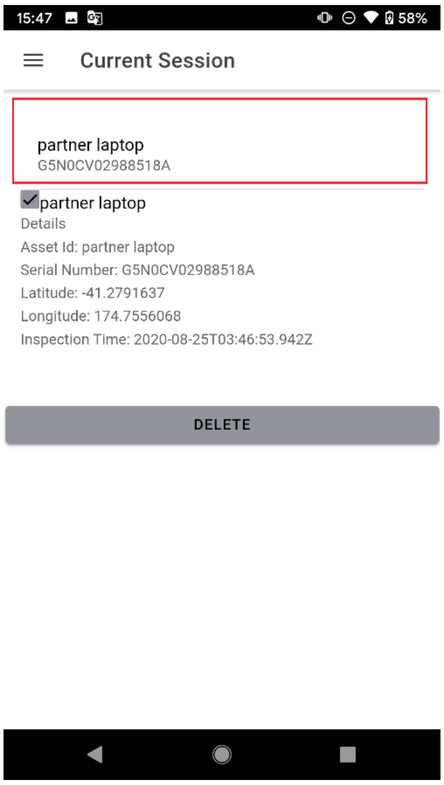


Figure 13: Navigation to Current Session

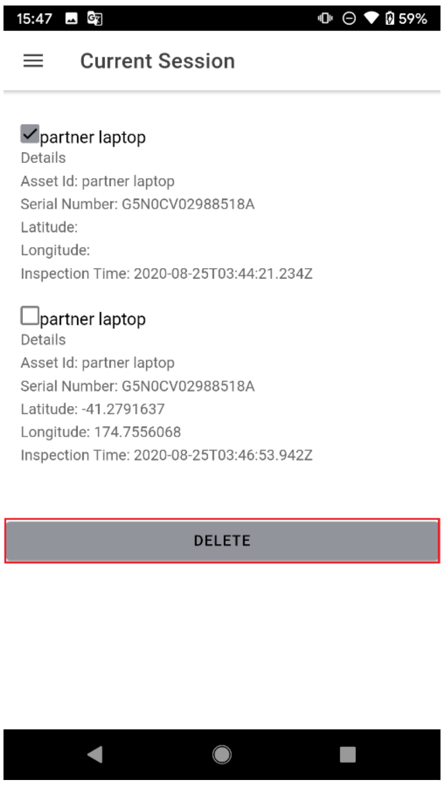
2. Individual items can be clicked on to view more information if needed



Details
Asset Id: partner laptop
Serial Number: G5N0CV02988518A
Latitude: -41.2791637
Longitude: 174.7556068
Inspection Time: 2020-08-25T03:46:53.942Z

Figure 14: Showing the current session and both an expanded (and selected) asset inspection, and a highlighted minimal overview of an asset inspection

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

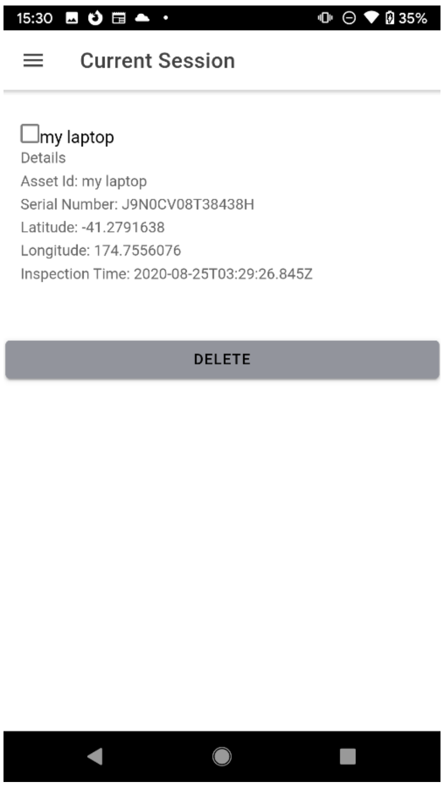

15:47 59%

Current Session

☑ partner laptop
Details
Asset Id: partner laptop
Serial Number: G5N0CV02988518A
Latitude:
Longitude:
Inspection Time: 2020-08-25T03:44:21.234Z

☐ partner laptop
Details
Asset Id: partner laptop
Serial Number: G5N0CV02988518A
Latitude: -41.2791637
Longitude: 174.7556068
Inspection Time: 2020-08-25T03:46:53.942Z

Figure 15: Duplicate with less information found, click delete to remove from the Current Session


15:30 35%

Current Session

☐ my laptop
Details
Asset Id: my laptop
Serial Number: J9N0CV08T38438H
Latitude: -41.2791638
Longitude: 174.7556076
Inspection Time: 2020-08-25T03:29:26.845Z

Figure 16: Current Session with information

Upload current session asset inspections to cloud

1. Navigate to upload to cloud

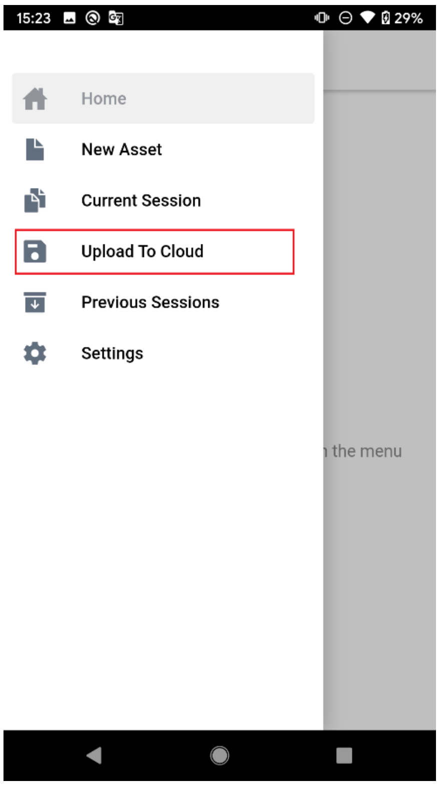
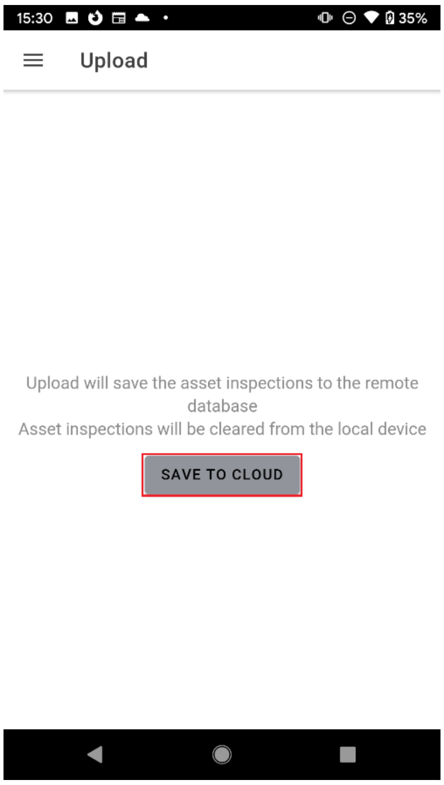


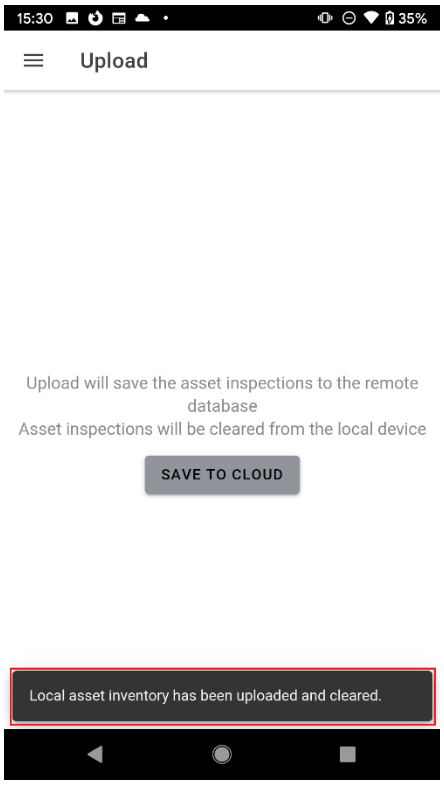
Figure 17: Navigation to Upload to Cloud

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



Upload will save the asset inspections to the remote database
Asset inspections will be cleared from the local device

Figure 18: Save To Cloud button to save to cloud and clear asset inspections from the local device



Upload will save the asset inspections to the remote database
Asset inspections will be cleared from the local device

Local asset inventory has been uploaded and cleared.

Figure 19: Toast to notify user that the upload was completed successfully

(Optional) Review previous sessions

1. Navigate to previous sessions

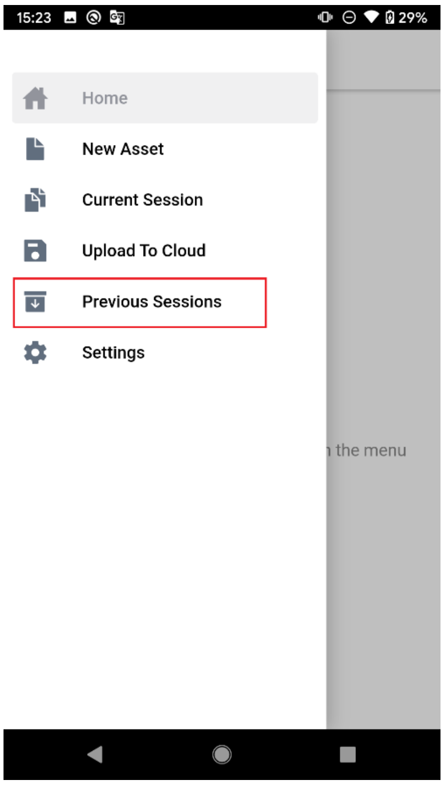
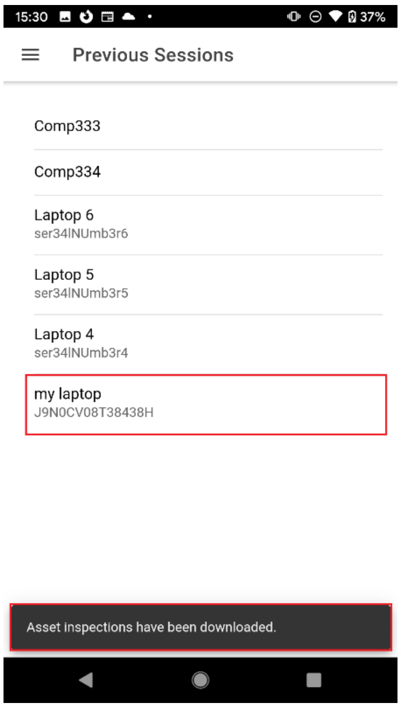


Figure 20: Navigate to Previous Sessions

2. Click on any asset inspection to see more details


15:30 37%

Previous Sessions

Comp333

Comp334

Laptop 6
ser34lNUmb3r6

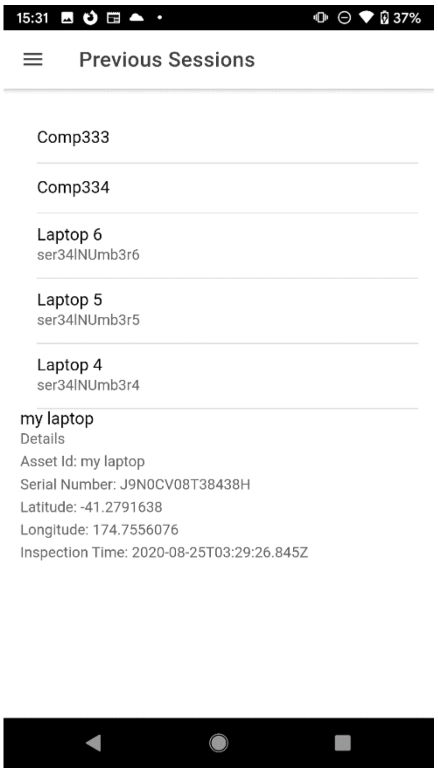
Laptop 5
ser34lNUmb3r5

Laptop 4
ser34lNUmb3r4

my laptop
J9N0CV08T38438H

Asset inspections have been downloaded.

Figure 21: Toast alerting the user that the Asset Inspections have been downloaded and a highlighted element that can be opened to review more details


15:31 37%

Previous Sessions

Comp333

Comp334

Laptop 6
ser34lNUmb3r6

Laptop 5
ser34lNUmb3r5

Laptop 4
ser34lNUmb3r4

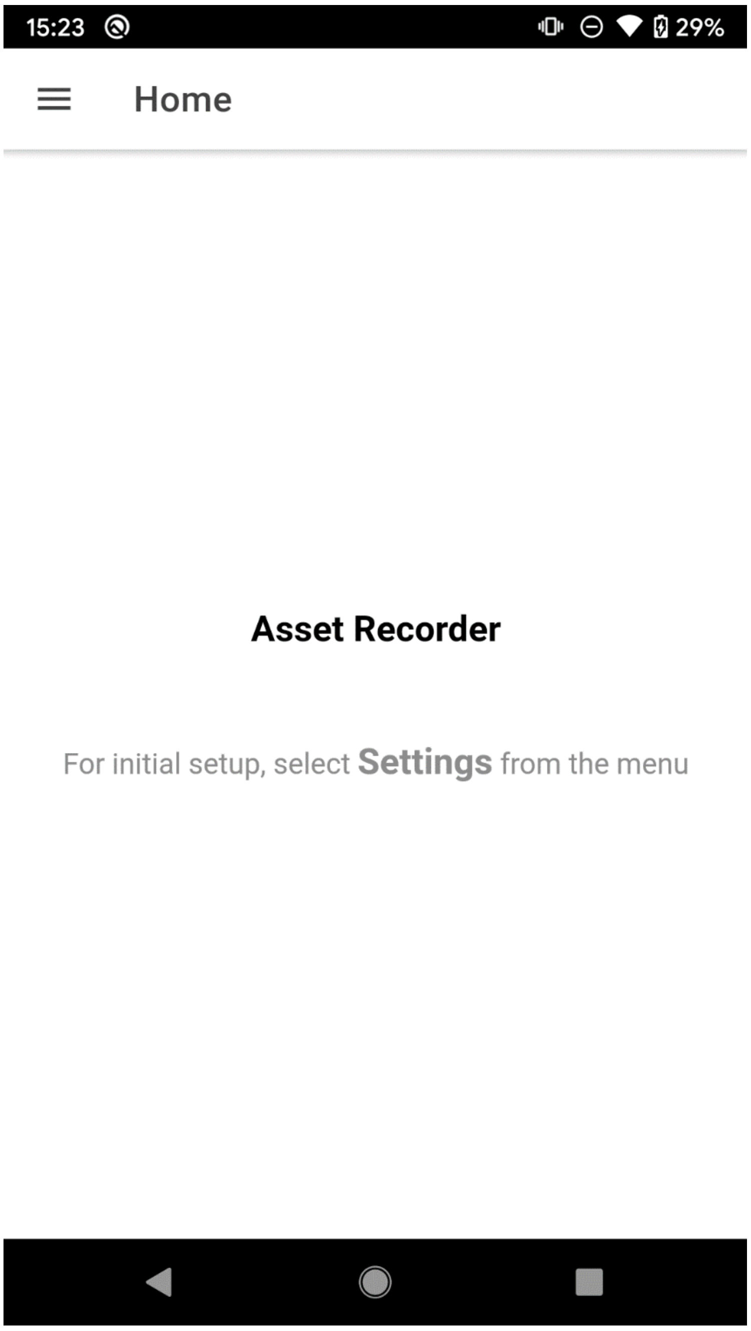
my laptop
Details
Asset Id: my laptop
Serial Number: J9N0CV08T38438H
Latitude: -41.2791638
Longitude: 174.7556076
Inspection Time: 2020-08-25T03:29:26.845Z

Figure 22: 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.

Home



Asset Recorder

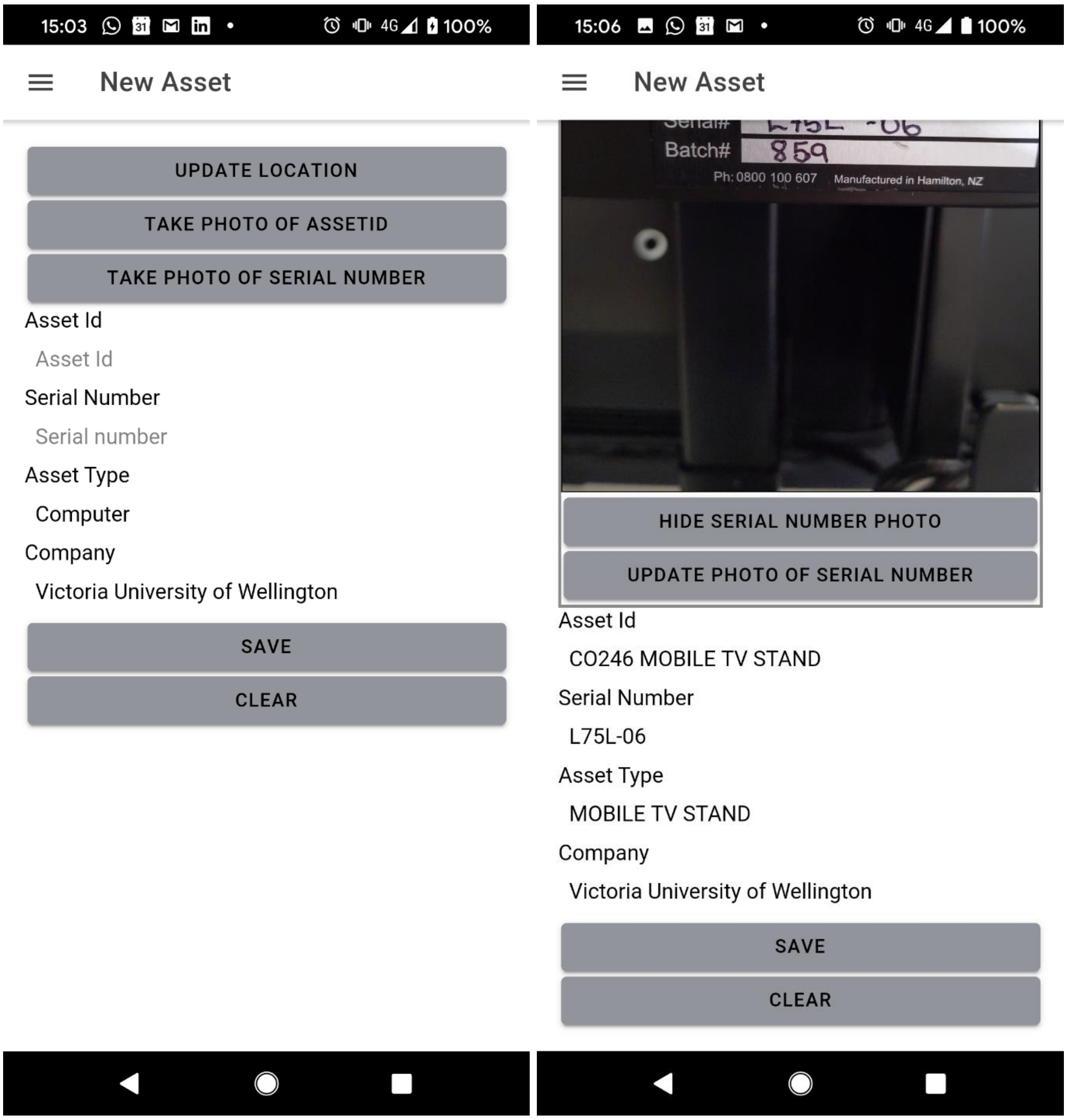
For initial setup, select **Settings** from the menu

Figure 23: Home screen which greets the user on opening the app.

Home screen is a simple screen that provides a hint for dealing with the initial setup of the application.

There is a hamburger menu up in the top left corner, a title of the app right in the centre and a muted hint for how to deal with using the app for the first time.

New Asset



Asset Id
Asset Id
Serial Number
Serial number
Asset Type
Computer
Company
Victoria University of Wellington

Asset Id
CO246 MOBILE TV STAND
Serial Number
L75L-06
Asset Type
MOBILE TV STAND
Company
Victoria University of Wellington

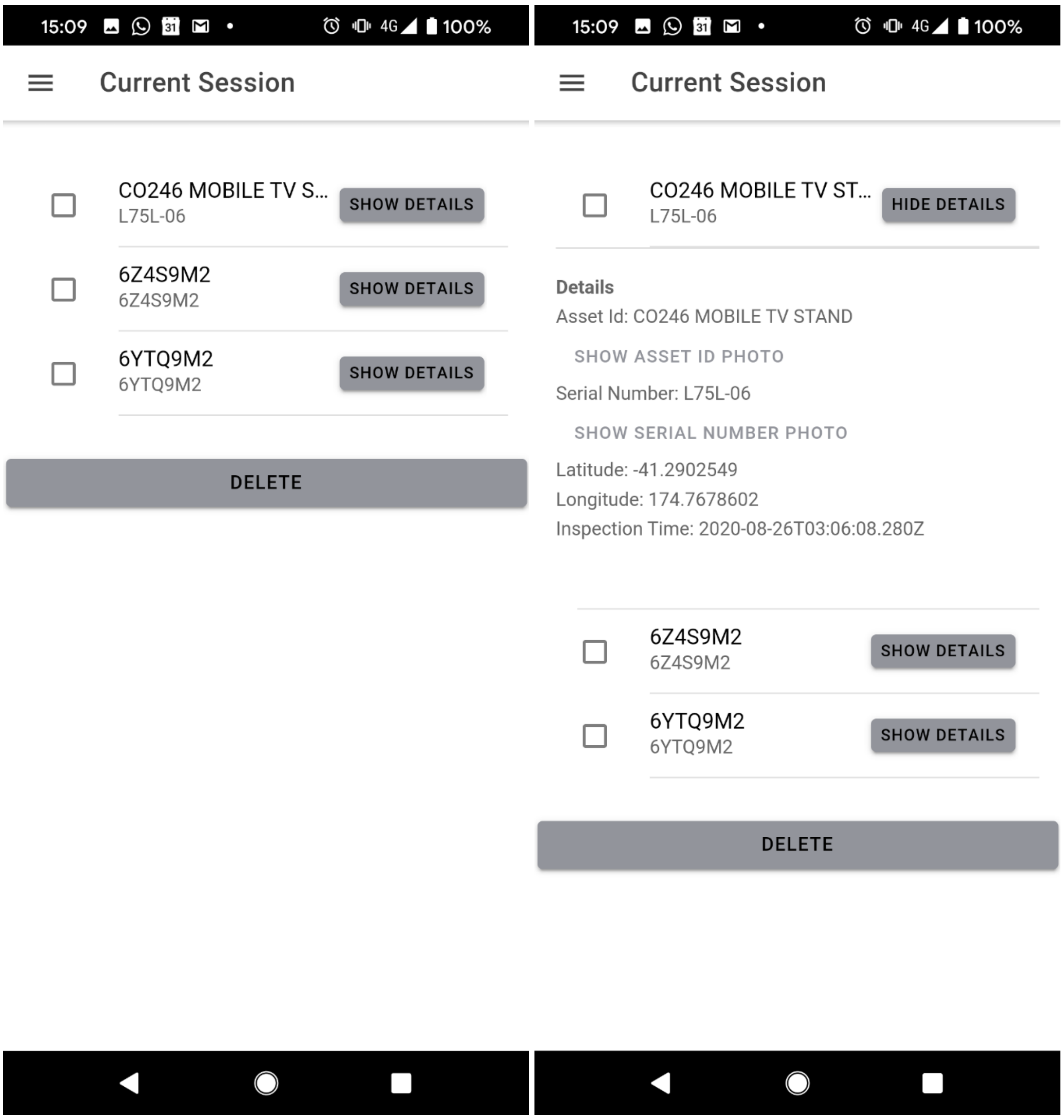
Figure 24: New Asset page, blank (left), filled (right)

The “New Asset” screen has buttons to update/set location, take photos of assets Ids and Serial Numbers, text boxes for the text of the asset Id and serial number and a prefilled asset type. The company is set in the settings but is displayed for users to view. There is a save and clear button to save/clear the current asset as needed.

When a photo is taken it is displayed with a hide and update button for the photo, and a border around the component to make it clear what controls are associated with the photo.

On save and clear, toasts will popup to let the user know that the asset inspection has been saved/cleared.

Current Session



15:09 4G 100%

Current Session

☐	CO246 MOBILE TV S... L75L-06	SHOW DETAILS
☐	6Z4S9M2 6Z4S9M2	SHOW DETAILS
☐	6YTQ9M2 6YTQ9M2	SHOW DETAILS

DELETE

15:09 4G 100%

Current Session

☐ CO246 MOBILE TV ST... L75L-06 HIDE DETAILS

Details

Asset Id: CO246 MOBILE TV STAND

SHOW ASSET ID PHOTO

Serial Number: L75L-06

SHOW SERIAL NUMBER PHOTO

Latitude: -41.2902549

Longitude: 174.7678602

Inspection Time: 2020-08-26T03:06:08.280Z

☐	6Z4S9M2 6Z4S9M2	SHOW DETAILS
☐	6YTQ9M2 6YTQ9M2	SHOW DETAILS

DELETE

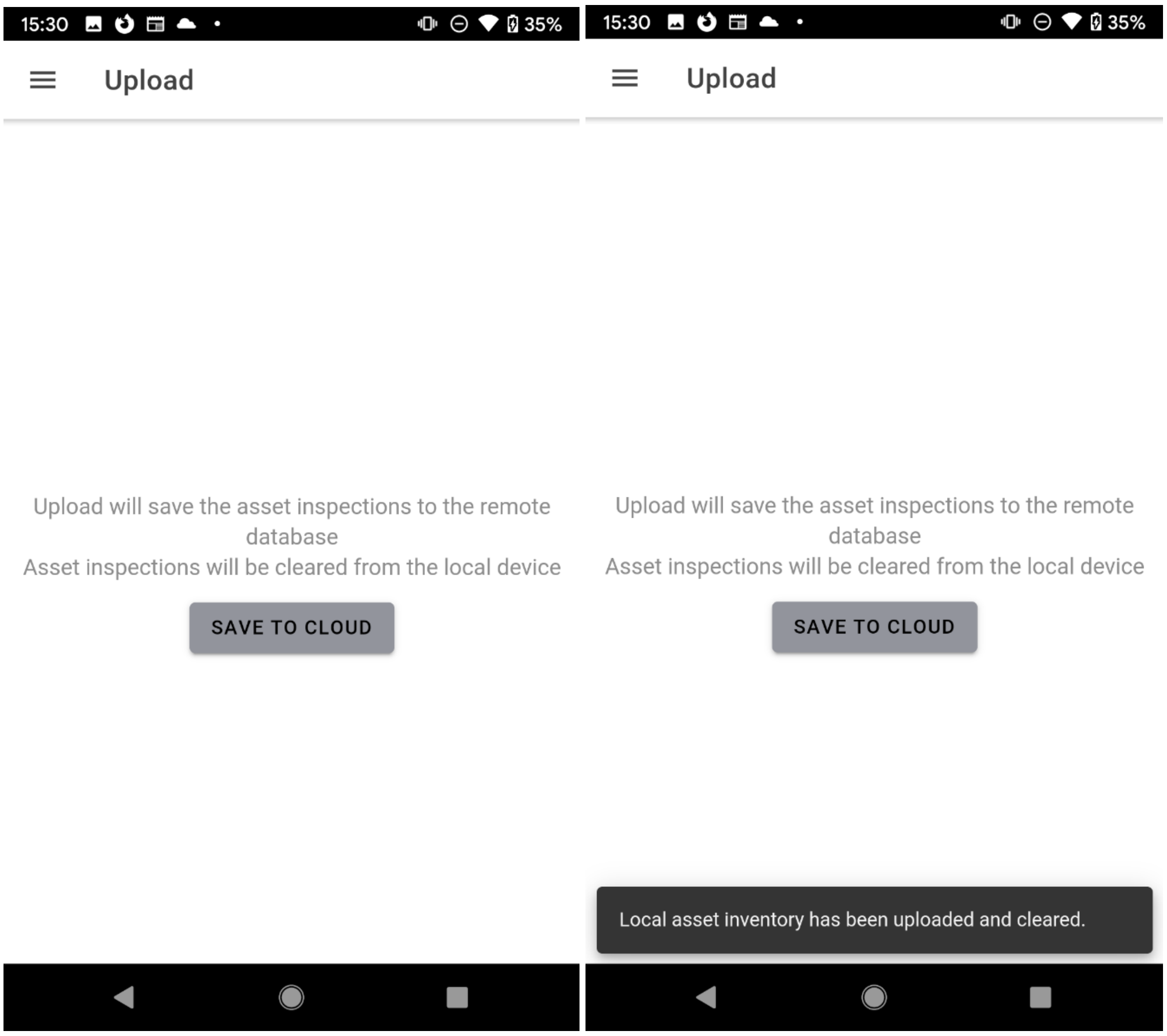
Figure 25: Current Session page, blank (no items expanded), one item expanded(right)

The “Current Asset” screen has an individual item for each of the recorded assets that have been saved in the “New Asset” screen. These assets have the asset ID and serial number shown, plus a “Show details button” and a checkbox for selecting items to delete.

The items details include buttons to show the associated photos and the button to show details changes to “hide details” when the details are expanded. The button remains in the same location making it easy to find and close the expanded details.

On selecting one or multiple items and deleting them, the screen reloads. This is because by default the item positions that were showing details continued to show details even if that element was removed (e.g. if the Mobile TV stand in position 1 was deleted, the next item (with asset Id 6Z4S9M2) would be in position 1 and have its details expanded). For the purpose of this assessment performing a full reload was seen as a reasonable work around, but not a great option.

Upload to Cloud



Upload will save the asset inspections to the remote database
Asset inspections will be cleared from the local device

Upload will save the asset inspections to the remote database
Asset inspections will be cleared from the local device

Local asset inventory has been uploaded and cleared.

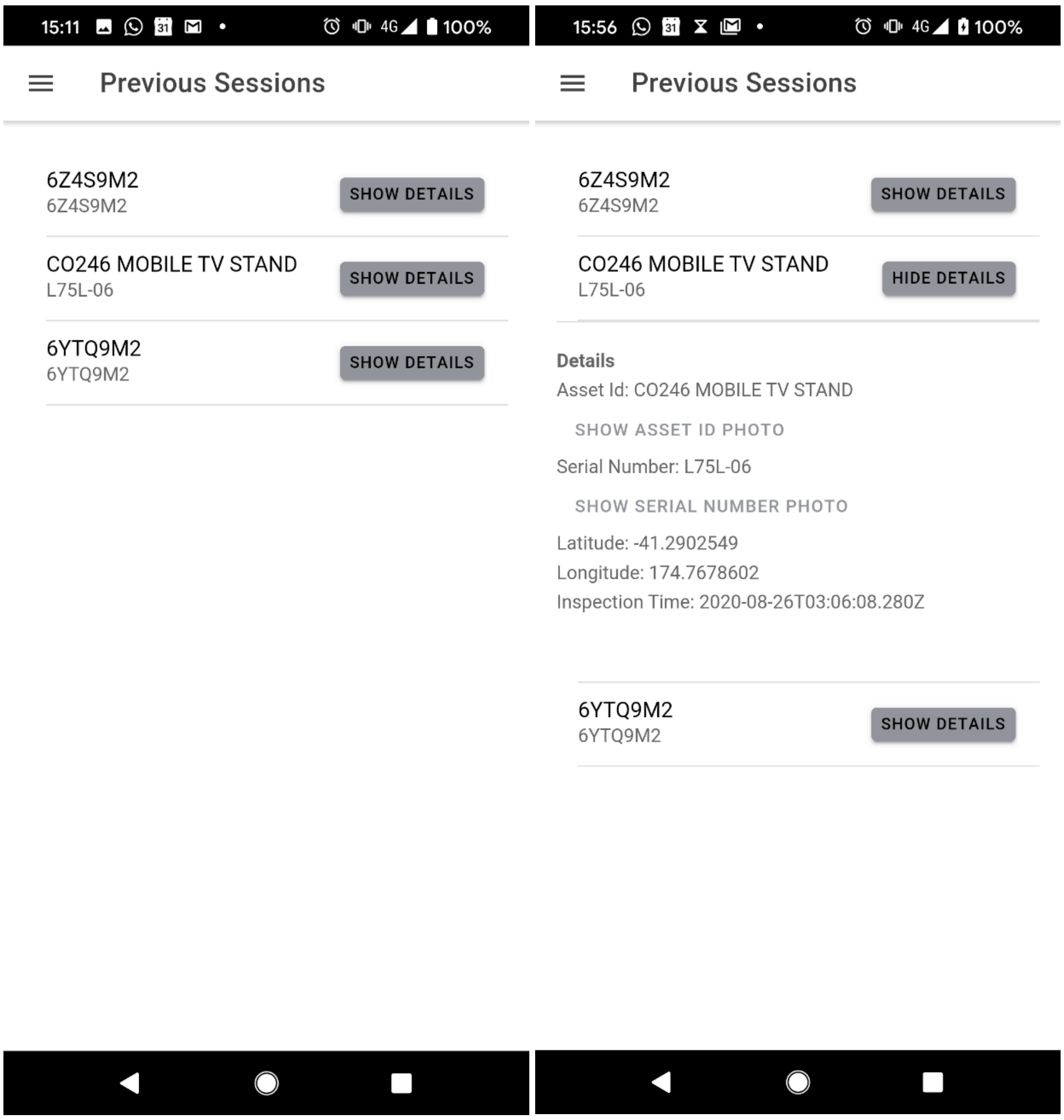
Figure 26: Upload page, before upload(left), after upload(right)

The upload to Cloud page is very minimal, with a single button and a muted tip about what uploading to the cloud will do to the information that is stored on the device.

Because information will be removed from the device and this action should only be performed as a conscious action it has its own screen to prevent mistakes or cluttering other screens.

On saving to the cloud, a toast will pop up and let the user know that the upload has been successful.

Previous Sessions


15:11 • 4G 100%

Previous Sessions

6Z4S9M2
6Z4S9M2
SHOW DETAILS

CO246 MOBILE TV STAND
L75L-06
SHOW DETAILS

6YTQ9M2
6YTQ9M2
SHOW DETAILS

15:56 • 4G 100%

Previous Sessions

6Z4S9M2
6Z4S9M2
SHOW DETAILS

CO246 MOBILE TV STAND
L75L-06
HIDE DETAILS

Details
Asset Id: CO246 MOBILE TV STAND
SHOW ASSET ID PHOTO
Serial Number: L75L-06
SHOW SERIAL NUMBER PHOTO
Latitude: -41.2902549
Longitude: 174.7678602
Inspection Time: 2020-08-26T03:06:08.280Z

6YTQ9M2
6YTQ9M2
SHOW DETAILS

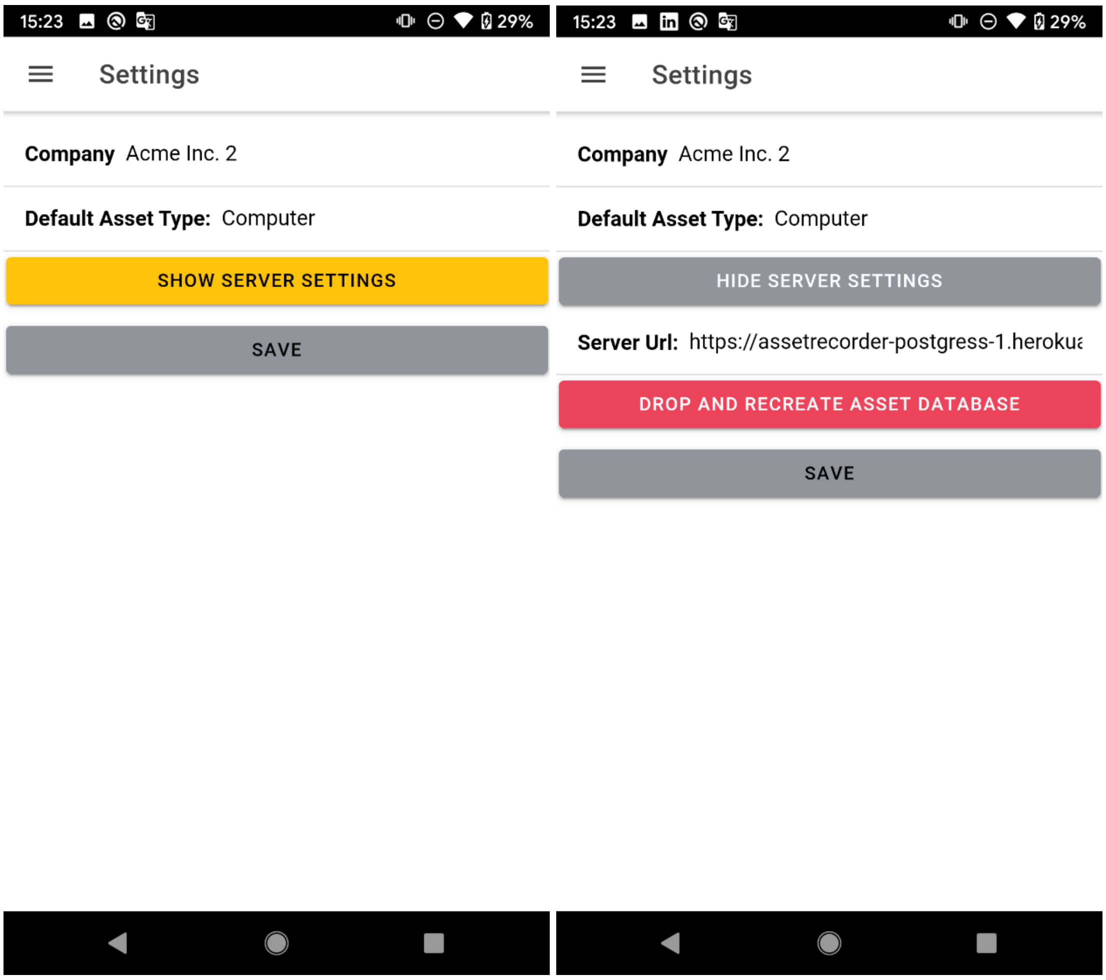
Figure 27: Previous Sessions page, before clicking to show details(left), after click to show details(right)

The “Previous Sessions” screen will pop up a toast with the message “Fetching inspections from remote server”. When the inspections have been downloaded, they will be displayed using the same format as the current session items, allowing for users to view the information and details. There is no delete button or check box to select to delete these inspections because they are not to be deleted as an inspection should be a read only object.

This provides a simple way to review all previously recorded and uploaded asset inspections, allowing a user to check what others have done and to ensure that their work has been uploaded.

The app is meant to be a way to collect information easily and quickly by users in the field, not a data entry/management tool.

Settings



15:23 29%

Settings

Company Acme Inc. 2

Default Asset Type: Computer

15:23 29%

Settings

Company Acme Inc. 2

Default Asset Type: Computer

Server Url: https://assetrecorder-postgress-1.herokua

Figure 28: Settings page, before clicking show server settings(left), after clicking show server settings(right)

The settings page is meant to be very rarely viewed, but also viewed on the first time using the application. Through this page the user can set the company and default asset type, change the server url and drop and recreate the asset database.

The company setting will be tagged on all of the asset inspections and there is no place to change it except in the settings, this is because it is unlikely there will be a change in company when performing asset inspections. The company setting is the main reason users are pushed toward settings from the home page for initial setup, but any of the other settings may also need to be changed.

The server settings are potentially very dangerous, and so are hidden behind a warning themed button. Within the server settings a user can drop and recreate the database which would destroy all data in it and is potentially very harmful, for this reason it has a danger theme. The show and hide server settings are kept in the same location to make navigation simple.

The save button is the same as the new asset save button.

Menu

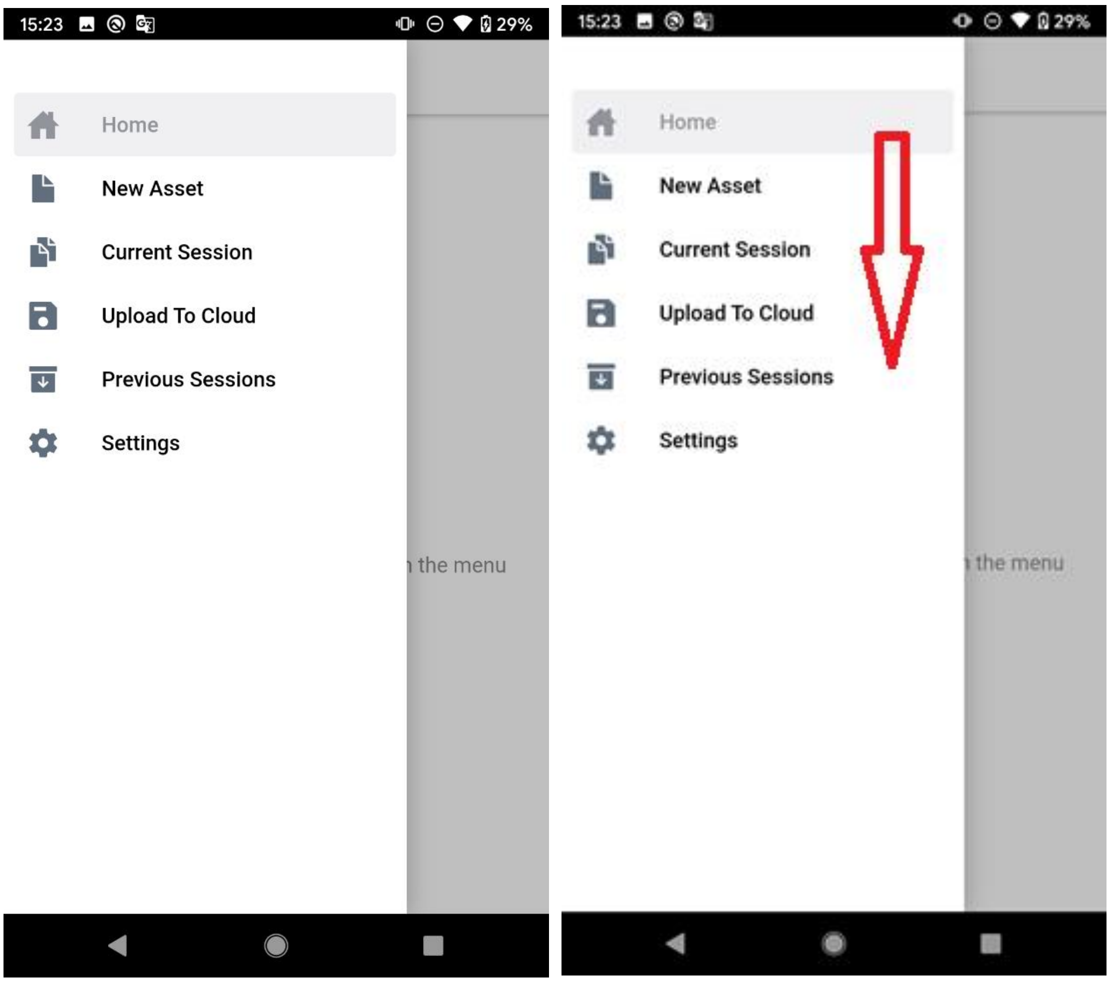


Figure 29: Menu(left), task flow (right)

While not a specific screen the menu in this application is one of the most used screens, allowing users to transition from one screen/task to another. The menu has icons which are ordered based on the expected task order, but allow for people to jump between screens/tasks as needed. As all navigation happens through the menu there is no confusion about how to access different screens to perform different tasks.

The icons have a consistent theme, going from a single sheet for New Asset, to multiple sheets for the current session, upload to cloud as a save item and previous sessions looking like an archive box which would be filled with sheets. Home, upload to cloud and settings may be seen as breaking the theme but each of them either is quite explicit in what it does or makes sense in the context of the other icons.