In addition to this application document, please see the wireframes at <https://pr.to/PISLO5/> (desktop host) and <https://pr.to/CDI4HP/> (phone/tablet client).

# Expectations

* The application running on the host machine should be built using Electron, Angular.js, ngMaterial, Sequelize + SQLite, and use Socket.io for communication.
  + Please see the Sequelize models located at <https://github.com/clarabyte/common-sequelize-models>.
  + Compatibility must be kept with the models we supply. Any additional models, columns, relations, helper methods, etc, should be submitted as pull requests to our repository.
* The application running on the client device should be built using Cocoonjs, Angular.js, ngMaterial, and use Socket.io-client for communication.
  + A TCP connection can be established over USB using ADB on Android devices, and with libimobiledevice for iOS. A TCP connection is required for Socket.io communication.
  + Should have the option to connect over WiFi as well.
* The application on the host machine will run in kiosk mode (fullscreen) when “NODE\_ENV” is set to “production”.
* The application for the host machine must be tested and work under Linux, and should be installable as a Debian package (.deb).
  + The .deb should include libimobiledevice and adb included when building. See <https://github.com/clarabyte/libimobiledevice-builder> for building libimobiledevice.
* Both applications, and any additional wireframes, will conform to the Material Design guidelines at <https://material.io/guidelines/>.
* The phone app should be installable wirelessly (WiFi, Bluetooth), and over USB by the host application.
* We would like everything to work with all iOS devices (if this is not feasible, devices running iOS 6.1.6 or newer), and Android devices running Android 4.4 or newer.

# Host Application

Please see the accompanying wireframes at <https://pr.to/PISLO5/>.

## Navigation

* Lock
* Shutdown
* Technician
* New Report/Active Report
  + Erasure Report
  + Functionality Report
* Unit status
* Previous Reports
* Device Logs
* Metrics
* Settings

## Account Settings

* Let the technician change their name, username, and passphrase.

## Active Report

* Phone cards should be laid out in the pattern specified by the USB hub they are attached to if the hub has been configured.
* If a client has not been selected and the “start” button is pressed, it should show a modal dialog saying to please select a client.

### Erasure Report

* Licenses should not be consumed when an erasure fails.
* Erasure types should be “factory reset” and “full erasure”. If “full erasure” is selected, and the device does not support it, the device should be marked failed.

### Functionality Report

* Wireframes should be fairly explanatory.
* In the report options section, there should be a dropdown that shows the different test suites that can be selected.
* If a test suite is not selected before pressing “start”, it should show a modal dialog saying to please select a test suite.

### Imaging Report

* Not shown in wireframes.
* Phone cards should have a dropdown selector added to them to select from a list of images.
* Device images should be automatically selected if the image has filters matching the device.
* If there are multiple images available it should select the most recent image.
* If there are no images with filters matching the device, the image selection dropdown should show the available images that have no filters associated.
* Licenses should not be consumed when writing an image fails.
* If any devices do not have an image selected, and a technician presses the “start” button, it should show a modal dialog with the text “Some devices do not have images selected. Start the report anyway?” With the options “no” (default), and “yes”

### Multiple Step Report

* Not shown in wireframes.
* Should be a single report where you can step through functionality testing, imaging, and erasure.

## Previous Reports

* On the previous reports there is a fab menu that when clicked should expand. From here you should be able to manually author a report for item tracking, create combined reports, and save reports to PDF or XLSX.

### Viewing a Previous report

* Not included in wireframes
* In the bottom-right corner there is a fab menu that when clicked should expand. From here you should be able to save the report to PDF or XLSX, and print asset tags on labels.
  + Printed asset tags should include roughly the same information as cards in the condensed mode.
* This should have two viewing modes; “condensed” and “full”
* The condensed viewing mode should show a card for each phone that shows the phone’s IMEI, serial, make, model, storage size, and status.
* The full viewing mode should show a table (see <https://github.com/daniel-nagy/md-data-table>), with a full list of device information (all information on the condensed mode card, plus RAM, processor, etc), and status information.
* For functionality reports:
  + The status on the condensed viewing mode should show a green check with the text “all tests passed” if all the applicable tests in the suite were passed when testing this device, otherwise it should show a list of failed tests.
  + The full viewing mode should show each test result in a column.
* For erasure reports:
  + The status on the condensed viewing mode should show the status of the erasure (passed, failed), and the type of erasure should also be shown on the card.
  + The full viewing mode should show the erasure status, and type of erasure.
* For imaging reports:
  + The condensed viewing mode card should include the image name, and whether it successfully wrote to the device.
  + The full viewing mode should show the same information as columns.

### Search

A filterable search box should be used here. Please note this is an element that will need to be created in a reusable way as it will be used in metrics and device logs as well, and we would like it to be made into a private bower component we can add to other tools.

### Filters

In addition to the following list of filters, we would like to be able to filter on any associated sequelize models.

* `is` - Filter on the enumerable status column.
* `after` - Filter on reports created on or after the supplied date.
* `on` - Filter on reports created on the supplied day.
* `before` - Filter on reports created on or before the supplied date.
* `client` - Filter on reports associated with clients matching the supplied name.
* `technician` - Filter on reports associated with technicians matching the supplied name.
* `serial` - Filter on reports that have a device with the supplied serial number.
* `imei` - Filter on reports that have a device with the supplied IMEI.

## Logs

* Not included in wireframes.
* Logging should include devices as they are connected; erasure and imaging start/stop/abort/finalize; technician sign in/sign out; edits to technicians, device images, clients, sites, etc.
* Forward hardware error status codes from client device to the host machine and log them. If an error status code applies to an action initiated from the host machine (functionality tests, erasure, rebooting a client, imaging, etc), some relation should be made between that action and the logged error for later lookup.

## Metrics

* Not included in wireframes.
* Time taken, error code frequency, devices run, machine, platform, pass/fail, etc.
* Data visualized with bar and pie charts with month by month drilldown and client and report connection visualization.
* A library like <http://www.chartjs.org> should make this easy.

## Settings

* USB hub configuration - Not included in wireframes
  + For initial configuration each hub should be plugged in one at a time after boot, and only each new hub should be shown.
  + After hubs are configured they should be stored in the database by serial number.
  + Configuration should allow the technician to set the rows and columns of the hub, and number order for each of the ports.
  + This will also need to take into account large hubs that appear as multiple hubs
  + It should be possible to reorder hubs in the hubs settings tab.
* Test suites
  + Configuration for test suites to enable/disable tests.
  + Should by default have three test suites:
    - All Tests
    - All Manual Tests
    - All Automated Tests
* Device images
  + A way to correlate which images should be used for reimaging a phone.
  + Filters should filter on data from ideviceinfo (in the example DeviceType is used), USB ID (vendorID:deviceID), and data from ADB.
  + New images will be added via a USB drive and need to be copied to the local “Imported Images” folder.
* Clients
  + Client name, info, etc.
  + Should have a default client named “Internal (no client)”.
* Technicians
  + Account usernames/passphrases and other information for each technician that may access the host machine.
* Sites
  + Local site is the current local site, and should be attached to a report for later retrieval by metrics.
  + Other sites may be created here.
* Import/export
  + Not included in wireframes.
  + Database management pane.
  + Should allow importing data from a database from a USB drive, and exporting to a database on a USB drive.
* Licenses
  + Not included in wireframes
  + This should let administrators add licenses for the application, which will be consumed individually in the following cases:
    - A test suite is ran on a device.
    - An erasure is ran on a device.
    - An image is written to a device.
  + See <https://github.com/clarabyte/registration-client>.
* About
  + In the future we’ll supply an EULA.
  + Clicking the “OPEN SOURCE LICENSES” button should display a list of libraries we depend on, and clicking on a library should show the open source license that it uses. The list should be ordered alphabetically, and there are tools for Node.js that can be used to pull all the license info for a given dependency. This should also include tools like libimobiledevice and ADB, which we include, but are not installed on the system.

# Android/iOS Application

Please see the accompanying wireframes at <https://pr.to/CDI4HP/>.

## Functionality Tests

* If phone is iCloud/Google Account locked and the app can not be installed: when running functionality report the phone should be marked automatically, with the option for a technician to mark that the app could not be installed for another reason.

### Manual Tests

The manual test wireframes should generally be self explanatory. Although the wireframes are sized for an iPhone, they should be designed to scale depending on the client device they are running on (both phones and tablets).

* These tests should be run as necessary provided a client device has the hardware available for testing. If a client device does not have the hardware required for a test then the test result should be marked “N/A”.
* Transitions between tests should be a horizontal slide from right to left over a 500ms duration using a quadratic ease-out function.
* Most manual functionality tests have one or two buttons at the bottom to allow the technician to mark the test as “passed” or “failed”. Tapping either button type should mark the test as indicated on the button, and move to the next test.
* Manual tests will typically have a 20 second delay before moving to the next test with the following exceptions:
  + Dead Pixels - When a technician presses the “start” button to begin the test, the countdown should pause until the screen finishes cycling between colors.
  + Any test that requires the technician to leave the app and return should pause the countdown.
* If a test is not passed within 20 seconds it should be marked as “timed out”.
* If the touchscreen tests fail, all subsequent tests should be marked as “untested”, as a technician will not be able to mark them as failed.

### Automated Tests

These tests should be run as necessary provided a client device has the hardware available for testing. If a client device does not have the hardware required for a test then the test result should be marked “N/A”.

* WiFi - Connect to a wireless access point on the host machine.
* Cellular - Can connect to a cellular network.
* Bluetooth - Can establish a connection to the host machine with Bluetooth.
* GPS - Can receive a GPS signal, and pull a location (latitude, longitude, altitude) from the client.
* Device is not jailbroken - Check to see that the client device is jailbroken or rooted.
  + One method would be checking to see if Cydia is installed on iOS devices, but other methods should be used to ensure this test is accurate.
  + This should also check to see if an Android device is rooted.
* Battery charge
* Vibration + Accelerometer - Read accelerometer and then pulse the vibration at different rates while continuing to read values from the accelerometer. Compare values to determine if both the accelerometer and vibration are working.
* Camera (back) + Flashlight - Take a picture from the rear camera with the flashlight off, then turn the flashlight on and take another picture. Average the lightness of each image.
* Camera (front) Read QR - Read a test QR code from the front camera.
* Camera (back) Read QR - Read a test QR code from the back camera.
* Microphone + Speaker - Play a number of different tones from the speaker and record the output. Analyze the recording to see if the tones were played by the speaker. This test should run with the volume set to 25%, 50%, 75%, and 100%, stopping when tones are detected and saving the volume level required before tones were detected with the report.
* USB Data - Verify that the device can transfer data over USB.
* USB Power - Verify the client device is receiving power over USB.
* IMEI Blacklist - If the device has an IMEI against global databases to see if the device’s IMEI is blacklisted.
* SIM Card - See if a SIM card is present. The test should fail if no SIM card is present.
* Activation Lock - See if the client device is iCloud locked or Google Account locked.
* Carrier Lock - See if the client device carrier locked.
* Warranty - See if the client device is within warranty.

# Additional

* We would also like to include a resource management solution for keeping iOS locked at a given version across multiple phones, as well as tracking possession.

## Definitions

* Host machine - The machine running the desktop app.
* Client/client device - The phone or tablet running the mobile app.