

[Description](#)

[Intended User](#)

[Features](#)

[User Interface Mocks](#)

[Screen 1](#)

[Screen 2](#)

[Key Considerations](#)

[How will your app handle data persistence?](#)

[Describe any corner cases in the UX.](#)

[Describe any libraries you'll be using and share your reasoning for including them.](#)

[Describe how you will implement Google Play Services.](#)

[Next Steps: Required Tasks](#)

[Task 1: Project Setup](#)

[Task 2: Implement UI for Each Activity and Fragment](#)

[Task 3: Your Next Task](#)

[Task 4: Your Next Task](#)

[Task 5: Your Next Task](#)

**GitHub Username:** Kureda

## Keep An Eye

### Description

#### **Problem:**

Your elderly parents want to be independent, but you worry that something might happen if you are not in touch. You want to be sure that they are OK, but you don't want to be constantly intruding all the time.

#### **Proposed Solution:**

Remote monitoring system, installed on their smartphones.

The app has two modes: "Client" and "Carer".

- App, installed in Client mode silently logs everyday activity and uploads it to the cloud.
- App, installed in Carer mode, downloads client's status and displays it.

(Prerequisites: users use their smartphones on everyday basis and have WiFi access)

## Intended User

Clients:

- Elderly person, living independently.
- People with special needs.
- People, recovering from recent illness.

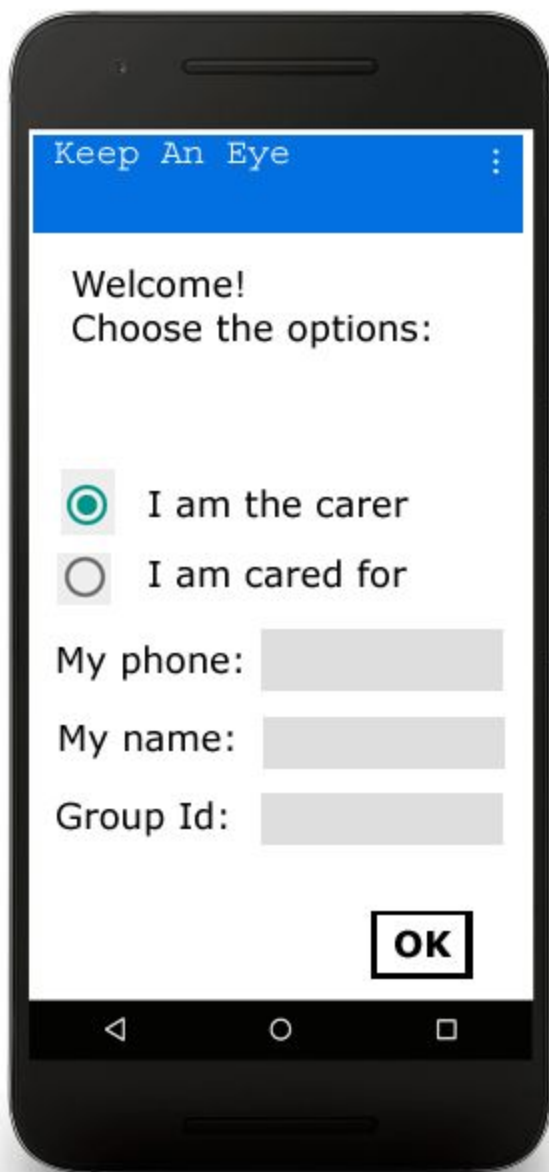
Carers: Relatives, friends or caregivers of the clients.

## Features

- At installation time user chooses mode: Carer or Client.
- Client monitors user's internet activity (unlocking smartphone) and physical activity (standing, walking, driving).
- Clients periodically uploads its status to cloud.
- Carer can monitor several clients.
- Carer periodically downloads all client updates from cloud and displays summary in a widget.
- On receiving update, the widget displays Green, Yellow or Red status.
- On clicking on the widget, the app opens and displays all client recent activity.
- Carer can send a message to a client or invoke a phone application.
- When new client installs, Carer displays it automatically.
- Carer can configure Client data (name, icon, phone number etc).

## User Interface Mocks

### Screen 1



Keep An Eye

Welcome!  
Choose the options:

☒ I am the carer  
☐ I am cared for

My phone:   
My name:   
Group Id:

OK

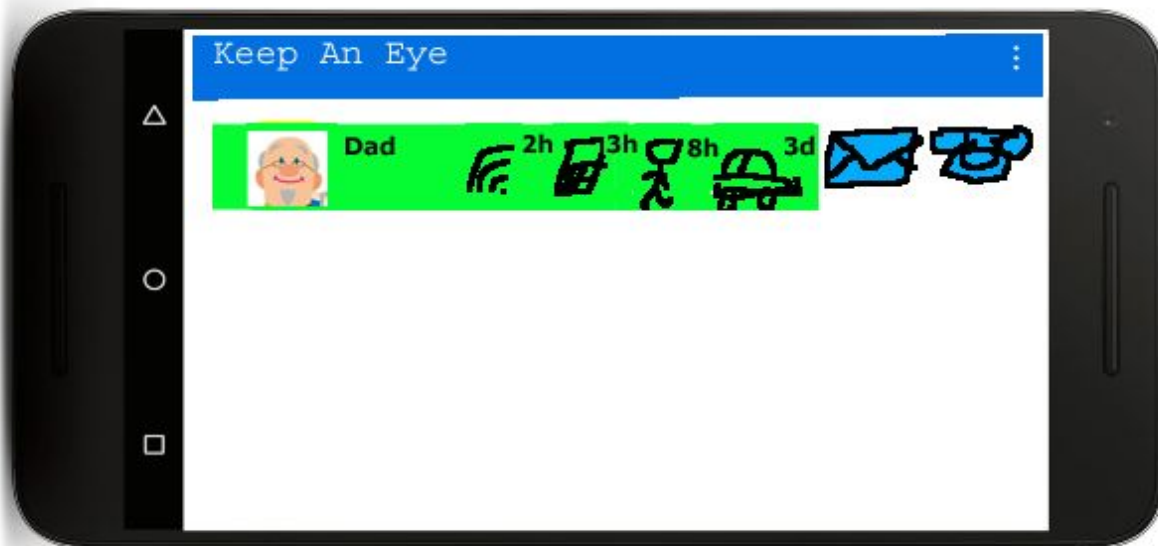
Screen for initial app installation. User chooses mode (Carer or Client) and enters required info.

## Screen 2



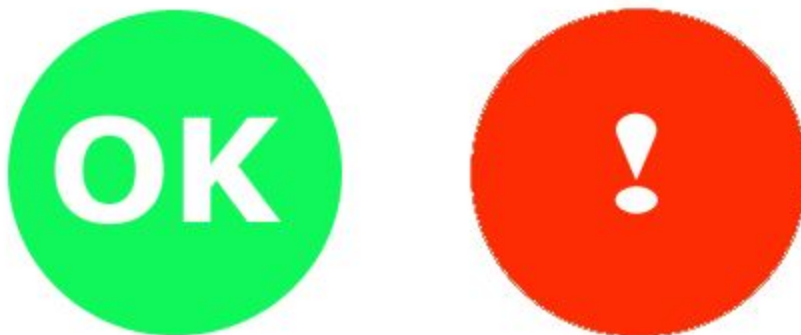
Carer main screen. Displays list of clients. In the picture Dad is OK, but Aunty haven't been using her phone for 8 hours and her phone is not uploading data for 8 hours! Something happened ! So her line is painted with red. The widget will also become red.

## Screen 2a



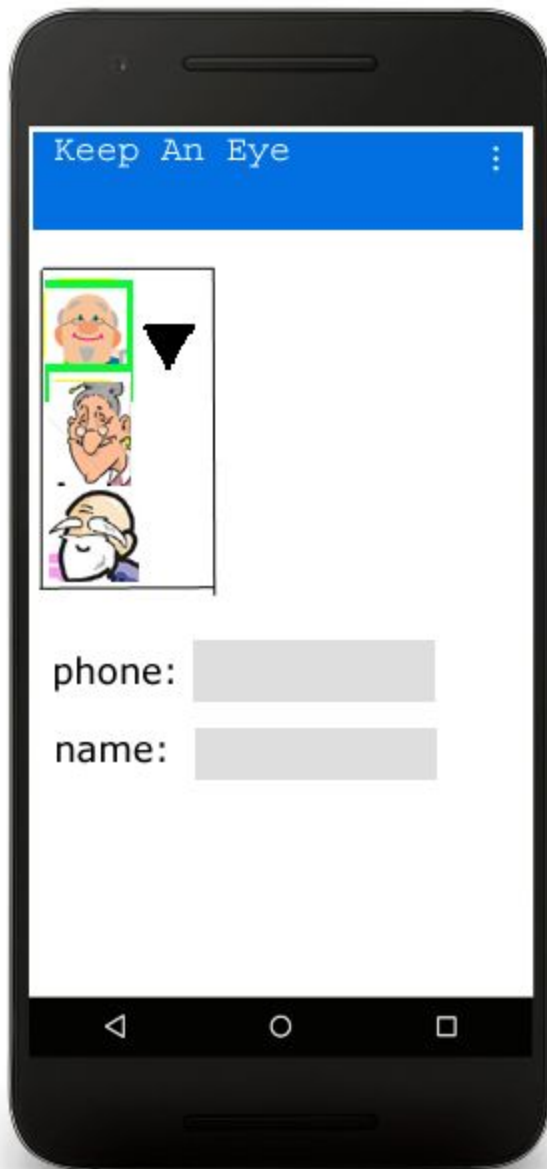
The same, but in portrait layout. Elements rearranged to stretch the row horizontally and shrink vertically.

## Widgets 2a



Widget is usually green, but if something happens, it goes yellow or red.

### Screen 3



Carer setting screen, to edit clients details: icon, displayed name and phone number.

## Screen 4



The main screen of cared mode. Simple layout, big details, to make it easy to call the carer or send them a message. Default message is displayed.

## Key Considerations

### How will your app handle data persistence?

- Client's recent activity is stored as Shared Preferences.
- Activity of all clients is stored as json file at myjson.com.
- Carer stores this info at a local database, wrapped at Content Provider and updates it with Sync Adapter.

### Example of json file:

```
{
  "groupId": "5265465456546",
  "members": [
    {
      "name": "Dad"
      "lastReport": "346235623454"
      "lastLogin": "3462398978978"
      "lastWalk": "34623234213"
      "lastRide": "34623111515"
    },
    {
      "name": "Mom"
      "lastReport": "34623567777"
      "lastLogin": "34623984444"
      "lastWalk": "346239999"
      "lastRide": "346222222"
    }
  ]
}
```

### Carer's database table structure:

```
name
phone_no
icon_id
when_last_updated
when_last_logged_in
when_last_drived
when_last_jogged
when_last_walked
```



### **Describe any corner cases in the UX.**

Concurrency case: several clients are trying to modify the json file simultaneously. Handled by implementing optimistic concurrency control.

### **Describe any libraries you'll be using and share your reasoning for including them.**

Schematics library. It makes creating Content Provider much easier.

### **Describe how you will implement Google Play Services.**

- Awareness API, uses Fences API, to monitor client's physical activity.
- Firebase messages, to send/receive messages from Carer to Client and back.

## **Next Steps: Required Tasks**

This is the section where you can take the main features of your app (declared above) and decompose them into tangible technical tasks that you can complete incrementally until you have a finished app.

### **Task 1: Choose visual design.**

- Color palette.
- Images, fonts, launcher icon.

### **Task 2: Create project stub.**

- InstallatinActivity
- ClientRecordingService
- ClientUploadingService
- CarerListClientsActivity
- CarerConfigureClientActivity
- ClientMainActivity

### **Task 3: Implement InstallationActivity.**

- Main view
- Landscape, tablet layouts, bundles
- Menus and navigation
- Functionality

#### **Task 4: Implement ClientRecordingService.**

- Make it wake up waked up by intents.
- Intent.ACTION\_USER\_PRESENT
- awareness.fence.DetectedActivityFence.WALKING from Awareness API
- Records last time of the event as Shared Preferences.
- Register in Google's Developer Console for the fence API events.

#### **Task 5: Implement ClientUploadingService.**

- Wake it up periodically (the period taken from config file)
- Download json file from myjson.com
- Include latest activity data into it
- Upload it back to the myjson.com
- Handle concurrency collisions (optimistic).

#### **Task 6: Implement Carer data layer.**

- Create local SQLite Database (table "Client")
- Create ContentProvider
- Create SyncAdapter
- Create CursorLoader

#### **Task 7: Implement CarerListClientsActivity.**

- Portrait view, with list of clients
- Loader loads data from Content Provider
- Actions
- Landscape, tablet layouts, bundles
- Menus and navigation
- Calculate status (Green, Yellow, Red), by activities and configs.
- Create Unit tests

#### **Task 8: Implement CarerConfigureClientActivity.**

- Portrait view
- Loader loads data from Content Provider
- Show name, icon, phone etc from settings
- Landscape, tablet layouts, bundles
- Menus and navigation
- Input validation
- Save settings to database
- Client removal

#### **Task 9: Implement CarerSummaryWidget.**

- Widget
- On rotation

- Display status (Green, Yellow, Red)

### **Task 10: Implement ClientMainActivity**

- Portrait view
- Landscape, tablet layouts, bundles
- Menus and navigation
- Implement messages sending (firebase messaging)
- Implement messages receiving

### **Task 11: Make the app more material**

- Shades
- Transitions with shared elements etc

### **Task 12: Add accessibility**

- RTL, content description
- Add second language (in strings.xml)

### **Task 13: Real life test.**

- Install app on group of smartphones.
- Test them for several days.

### **Task 14: Build.**

- Keystore, signing configuration
- Create group of smartphones for Udacity tester
- Put to github, submit to Udacity.

### **Task 15: Celebrate.**

- Wait for the reply and hope for the “Pass”.
- Print the certificate, hang it on the wall.
- Throw a party.

---

### **Submission Instructions**

1. After you’ve completed all the sections, download this document as a PDF [ File → Download as PDF ]
2. Create a new GitHub repo for the capstone. Name it “**Capstone Project**”
3. Add this document to your repo. Make sure it’s named “**Capstone\_Stage1.pdf**”