GitHub Username: @ashchuk (https://github.com/ashchuk)

Cuckoo: simple notification app

Description

We often use instant messengers to ask each other and anything.
And often it happens that we immediately forget that we have just read.

Cuckoo helps other people edit your task list and notify your friends about what you are currently doing.

When someone edits your todos list, you automatically receive a notification that you have a new todo task. The task will be added in the todo list without your direct involvement.

Select the appropriate status from the list of available and all users subscribed to you will receive a notification.

The Cuckoo application allows users to:

- Easily send each other notices;
- Share information about your location, and automatic and manual mode;
- Edit todo lists;
- Share status (at work, at home, on the road and other statuses) with friends;
- Request each other's GPS coordinates and each other's status;
- Automatically send notifications when the position changes using Geofences;

The application is conveniently managed from the lock screen using the widget.

Intended User

Drivers, busy people, married couple with babies

Features

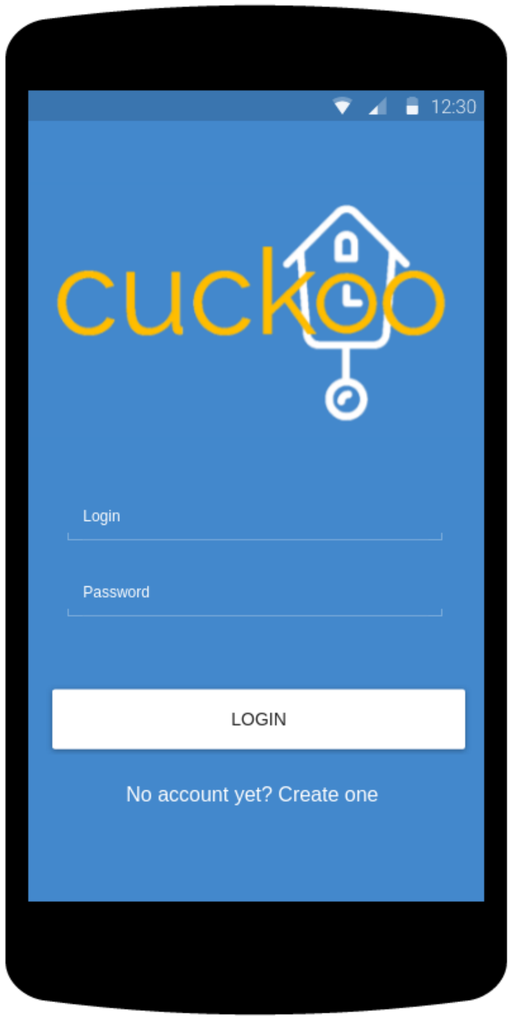
Main app features:

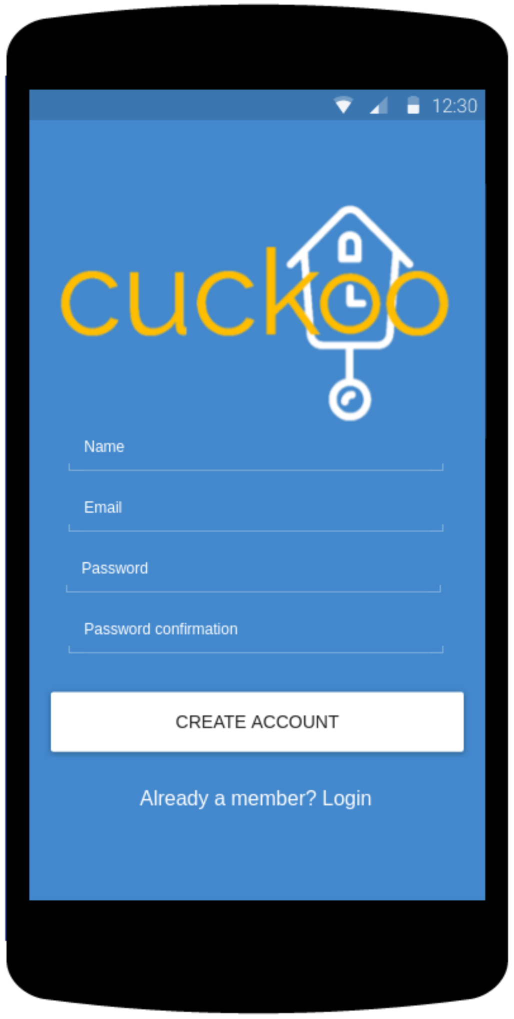
- Saves information
- Sending messages
- Show notifications
- Share GPS coordinates
- Geofences
- Share Todo sheet

User Interface Mocks

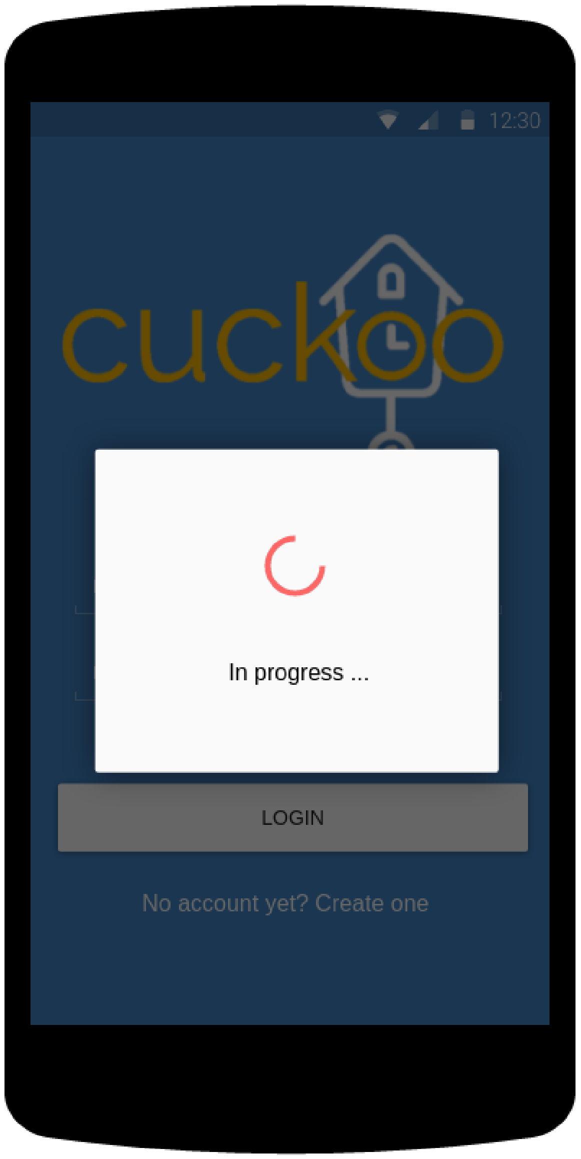
Screen 1

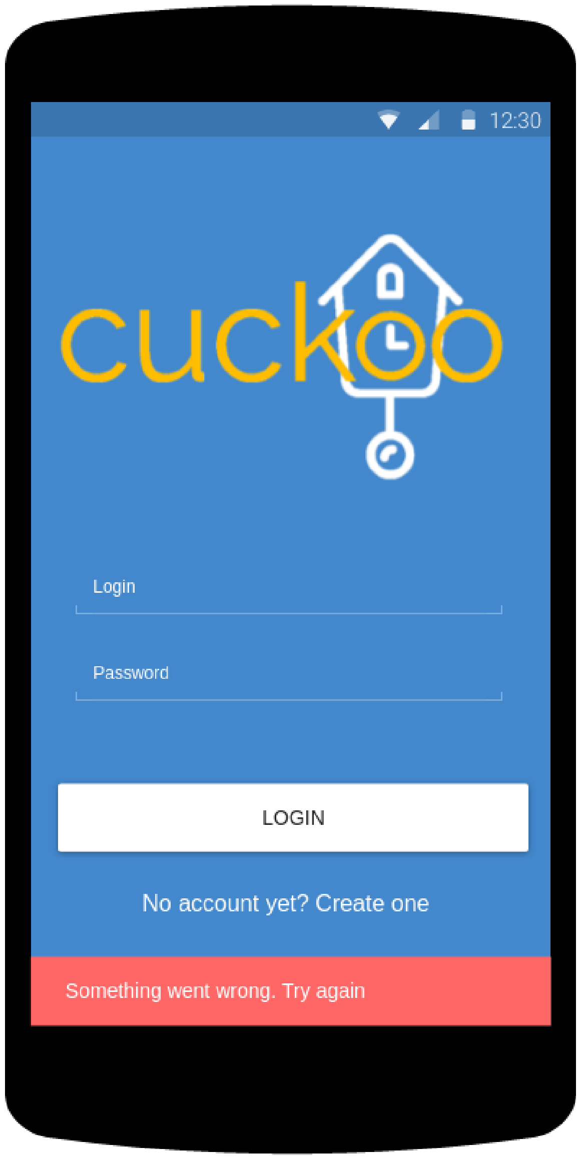
Login screen and create new account screen





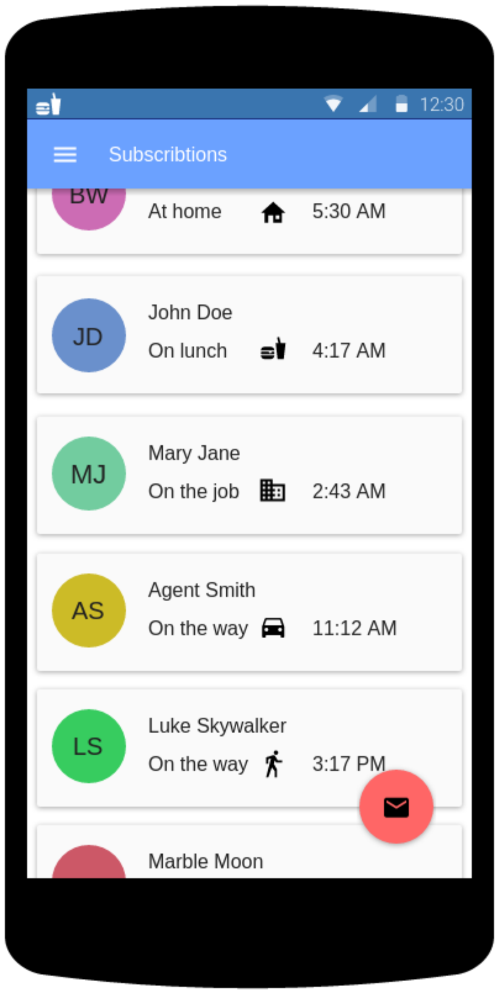
Login screen: loading dialog and error message showing.



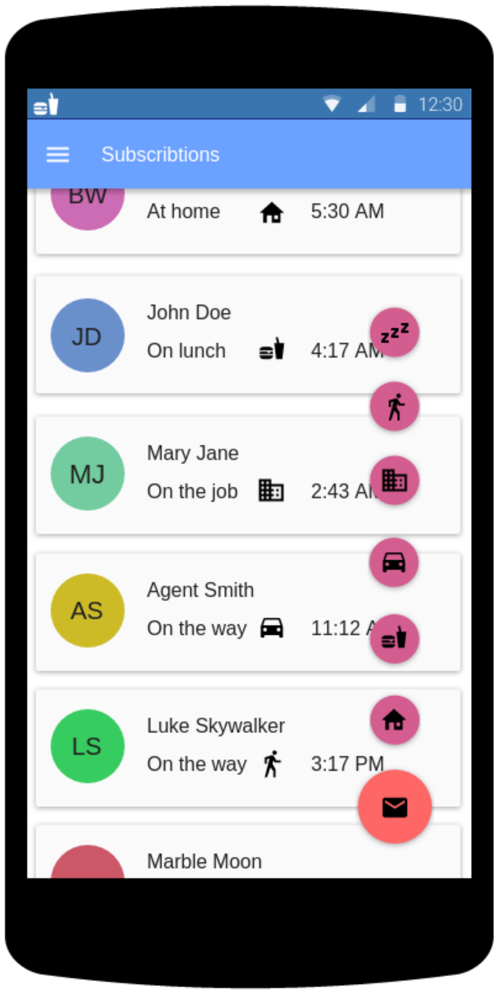


Screen 2

Subscribe page. Here user can see his subscribtions and change own status using FAB button. Then user press on list item an user details page opens.



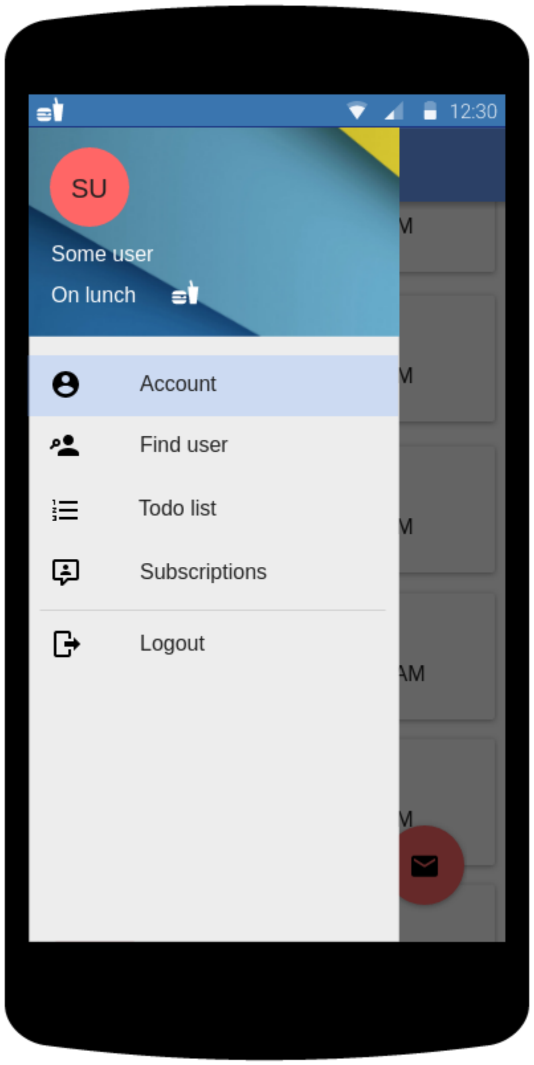
BW At home 5:30 AM
JD John Doe On lunch 4:17 AM
MJ Mary Jane On the job 2:43 AM
AS Agent Smith On the way 11:12 AM
LS Luke Skywalker On the way 3:17 PM
Marble Moon



BW At home 5:30 AM
JD John Doe On lunch 4:17 AM
MJ Mary Jane On the job 2:43 AM
AS Agent Smith On the way 11:12 AM
LS Luke Skywalker On the way 3:17 PM
Marble Moon

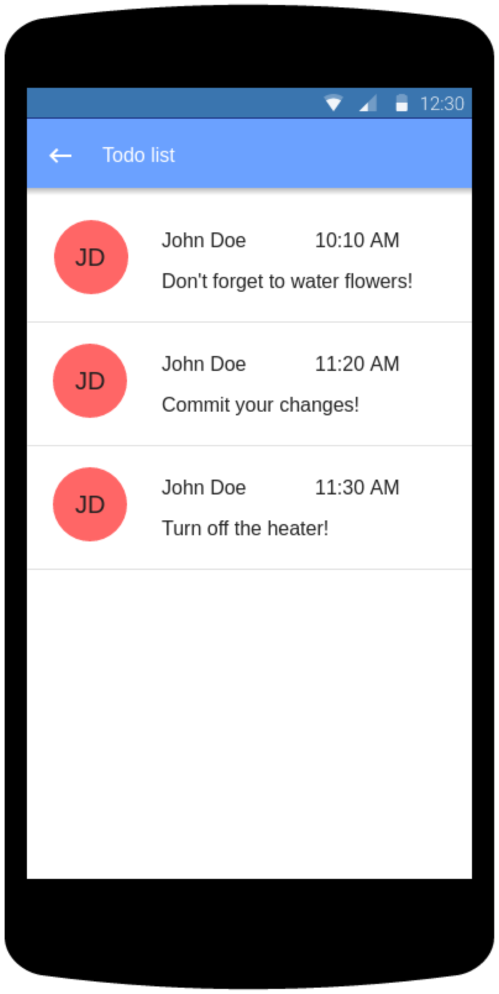
Screen 3

Slide menu



Screen 4

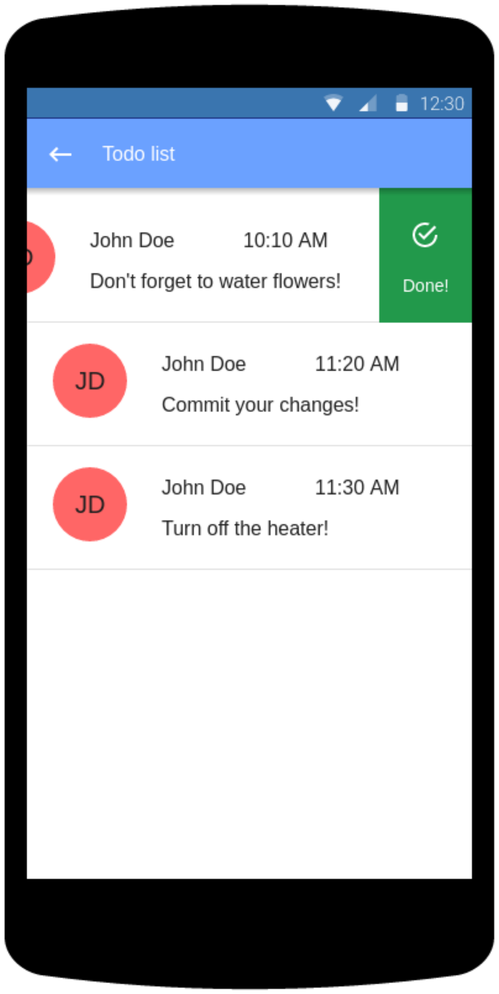
Todo list screen. Here user can check his todo list items and mark them as “Done”. Each item contains info about task issuer and task creation time.



John Doe 10:10 AM
Don't forget to water flowers!

John Doe 11:20 AM
Commit your changes!

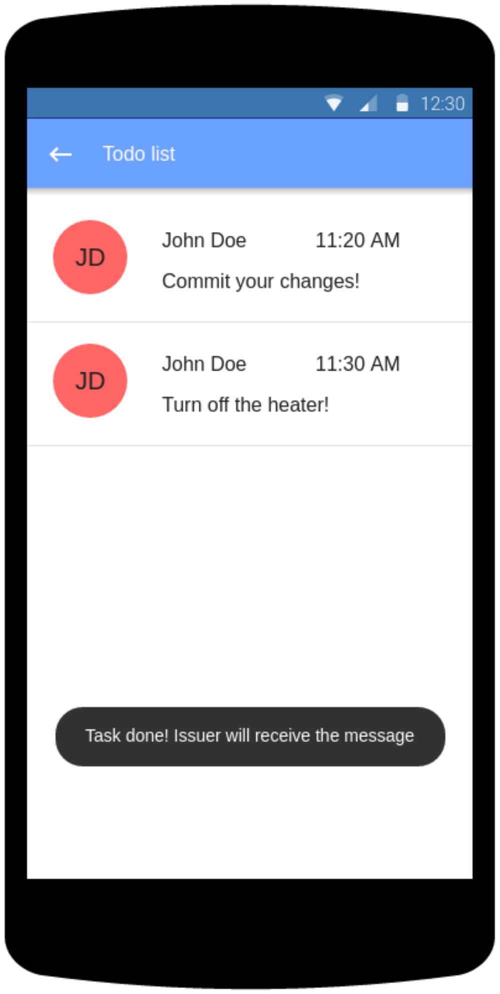
John Doe 11:30 AM
Turn off the heater!



John Doe 10:10 AM
Don't forget to water flowers!

John Doe 11:20 AM
Commit your changes!

John Doe 11:30 AM
Turn off the heater!



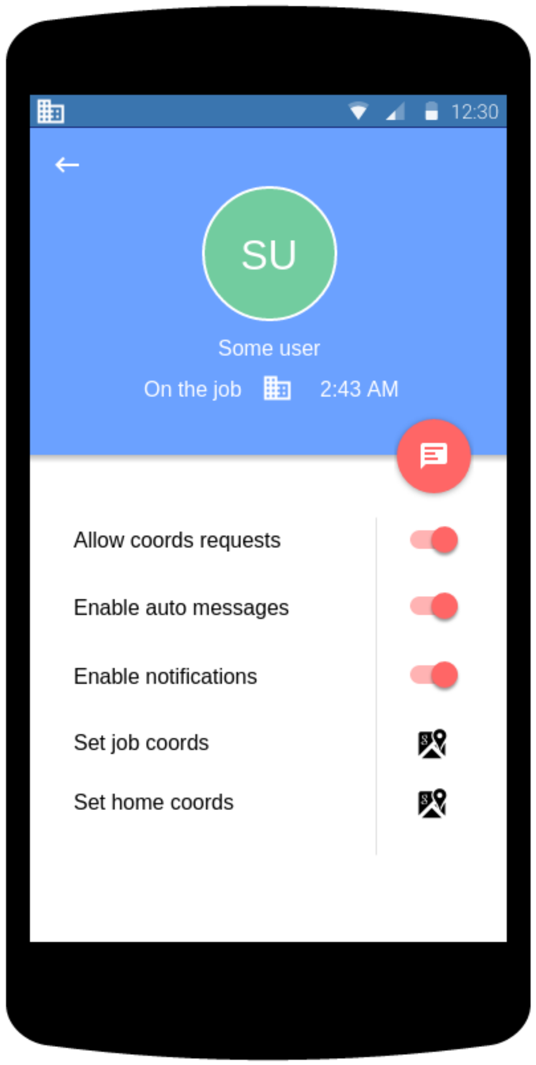
John Doe 11:20 AM
Commit your changes!

John Doe 11:30 AM
Turn off the heater!

Task done! Issuer will receive the message

Screen 5

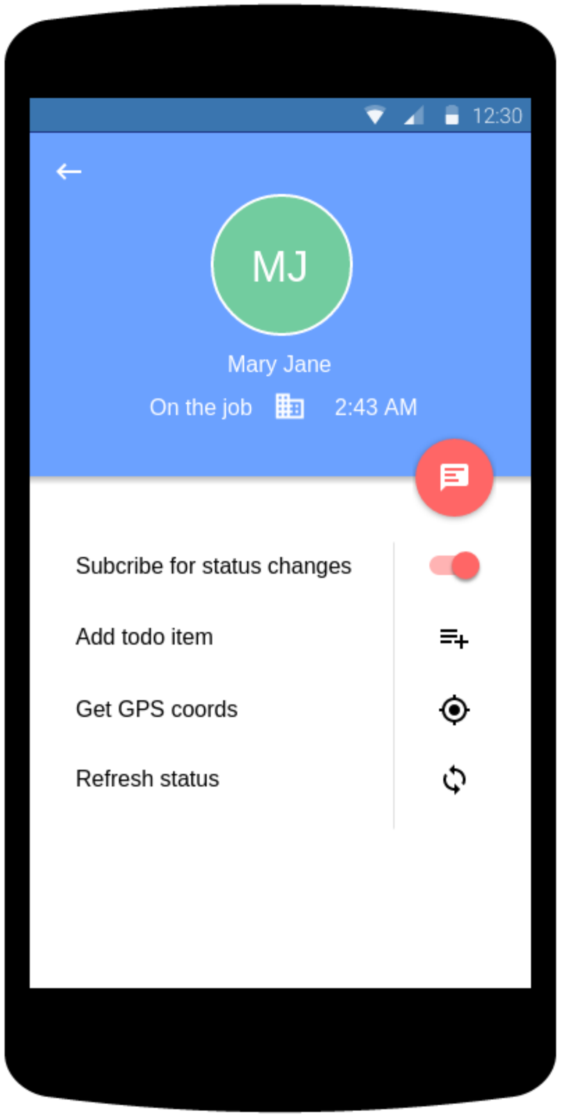
Account page. Also it’s a settings page. Also user can create a todo list item for himself by pressing a FAB button.

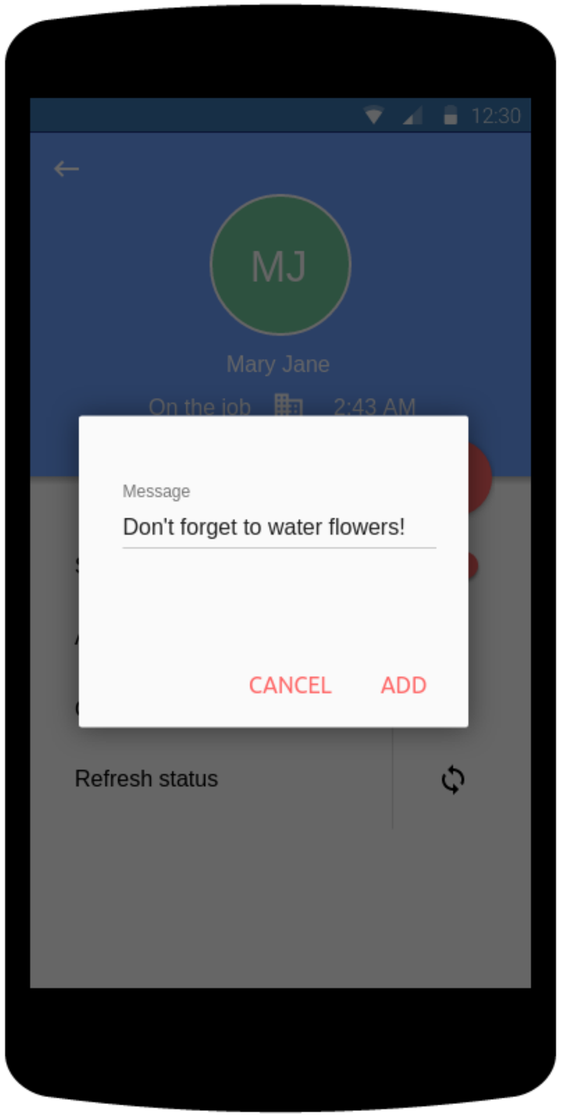


Allow coords requests
Enable auto messages
Enable notifications
Set job coords
Set home coords

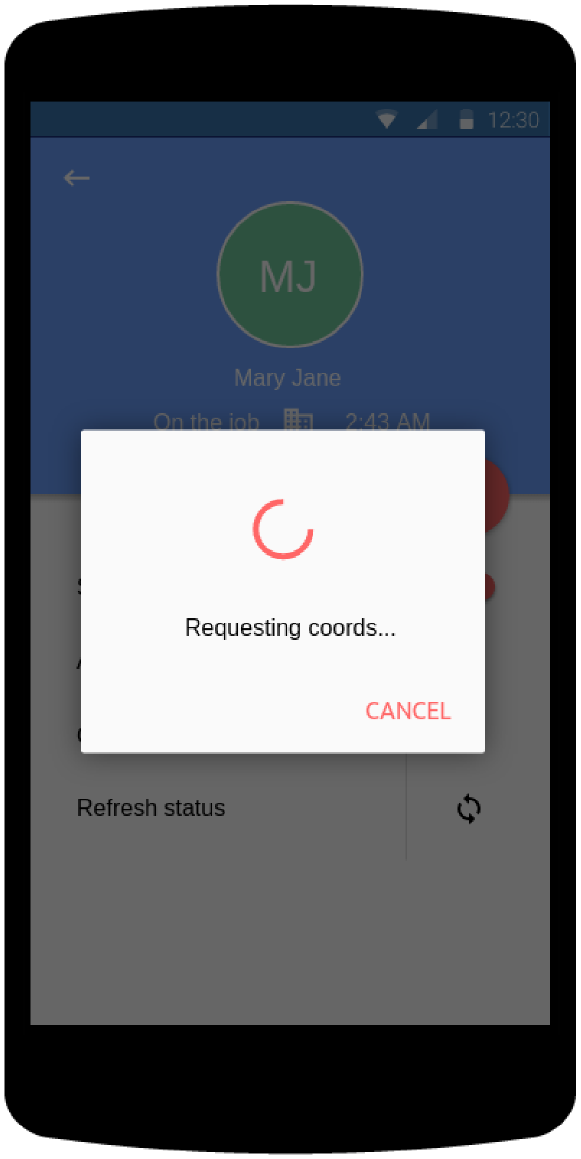
Screen 6

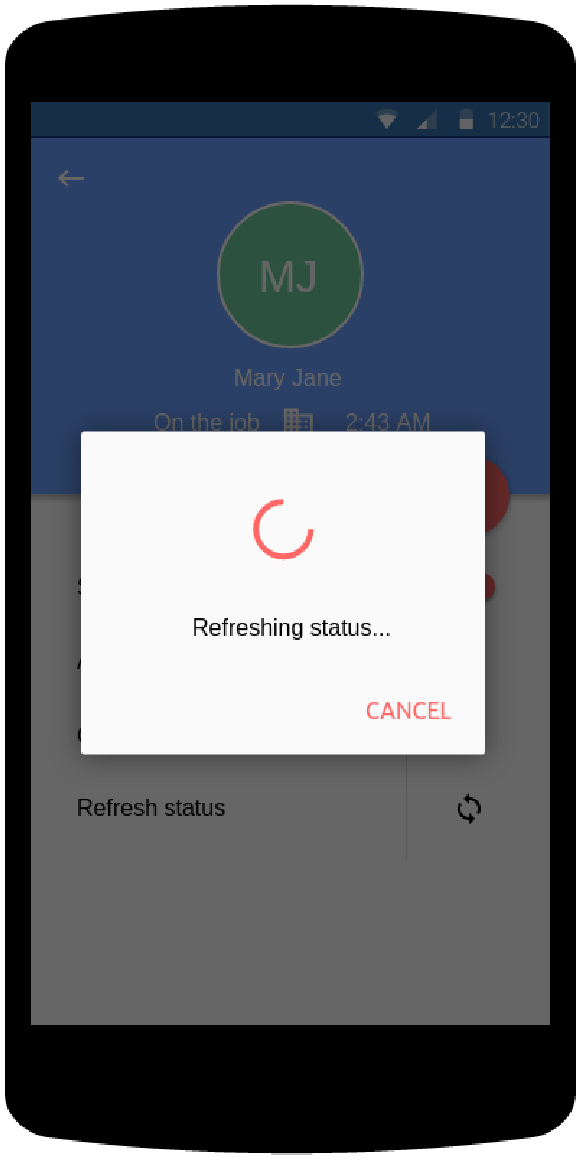
User details page. Here we go then pression on list item on subscribe page. Also there is a create todo list item dialog.





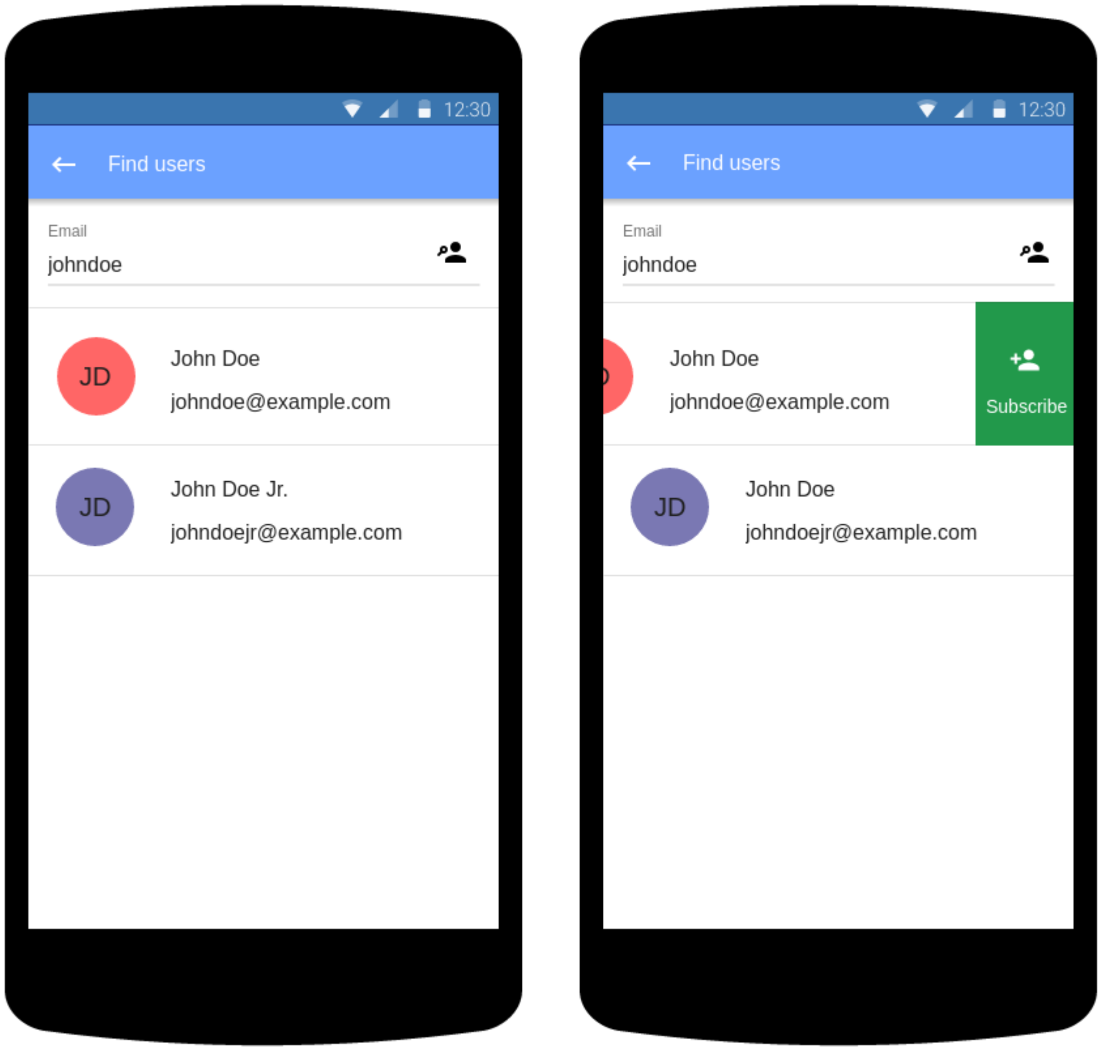
User details page. Here is two progress dialogs.



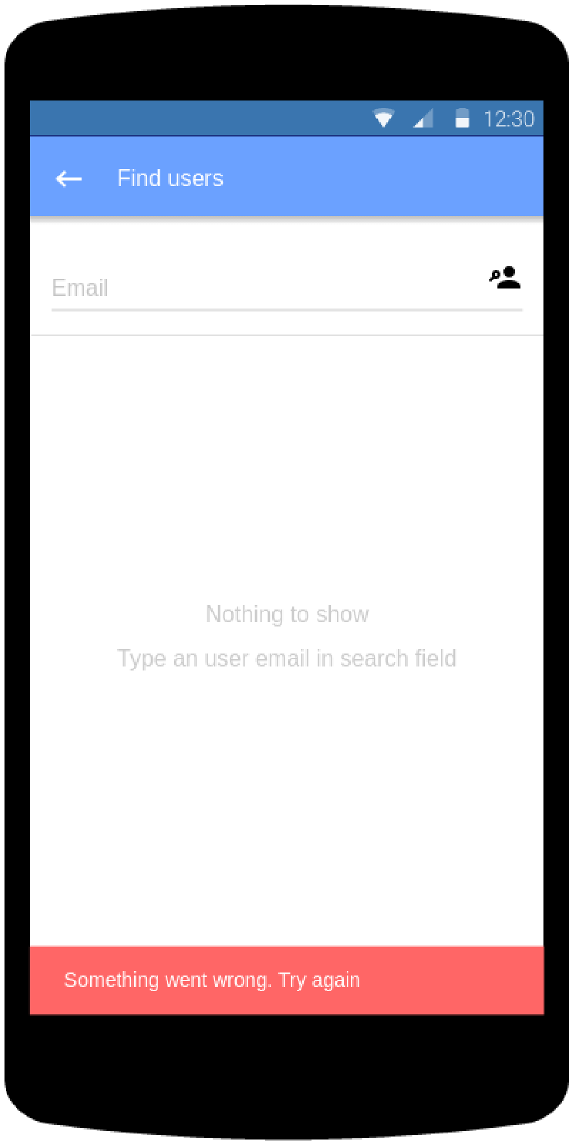


Screen 7

Search page. Here user can search for another users and create new subscribtions.



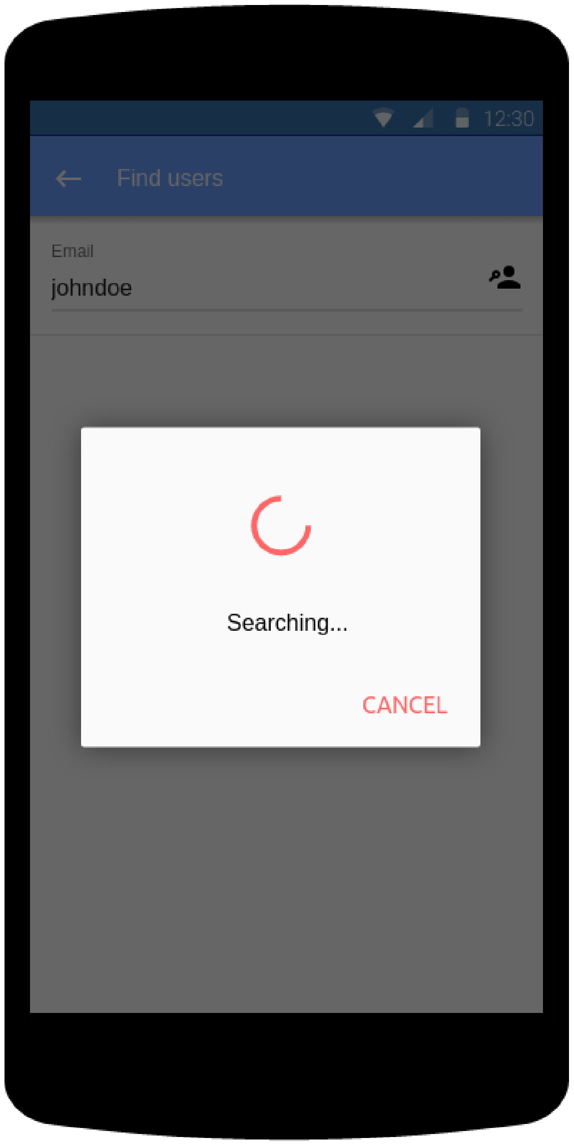
Search page. Here is an progress dialog, error message and empty list sample.



Nothing to show

Type an user email in search field

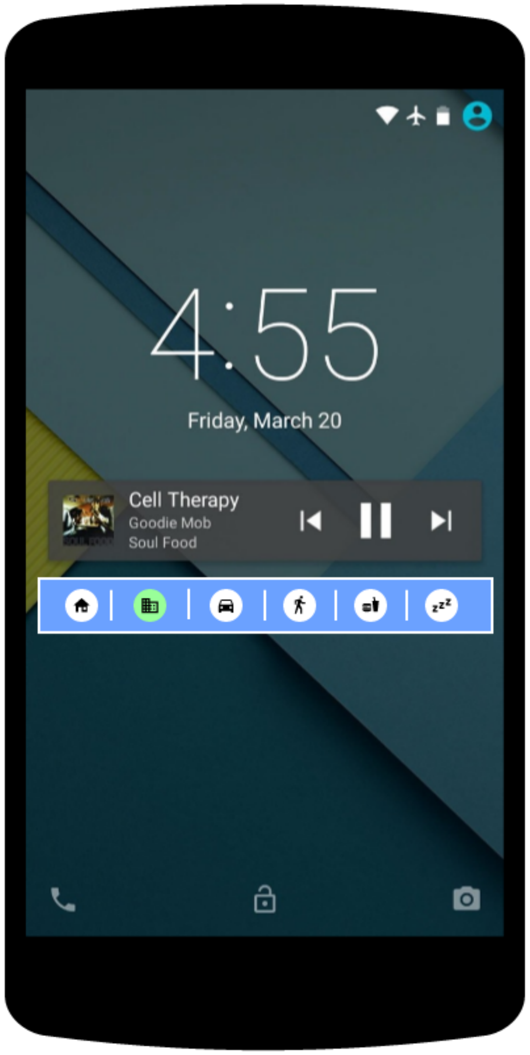
Something went wrong. Try again

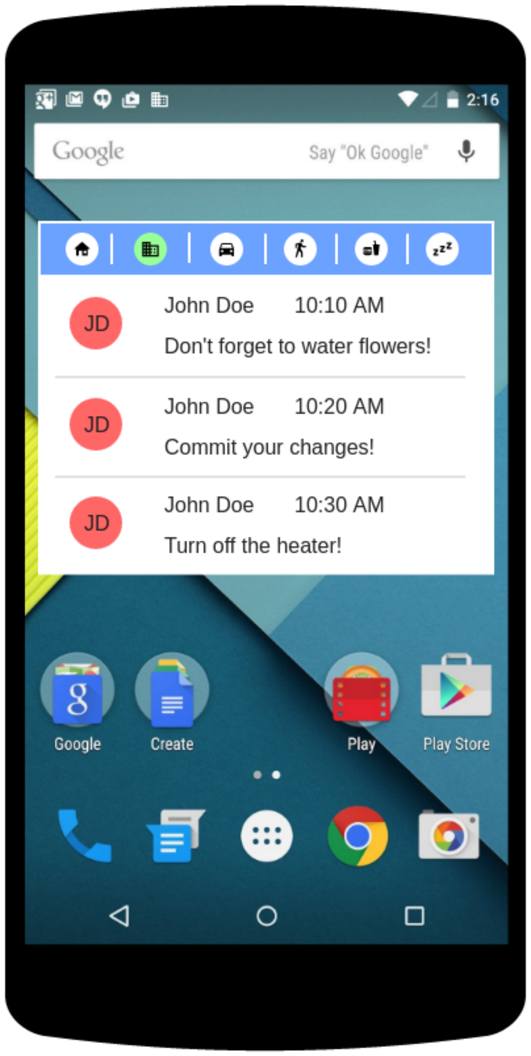


Searching...

Screen 8

Widget. Left image – lock screen widget, and right side – gome page widget. Current status is highlighted green, user can select another one by pressing any icon.





JD John Doe 10:10 AM
Don't forget to water flowers!

JD John Doe 10:20 AM
Commit your changes!

JD John Doe 10:30 AM
Turn off the heater!

Key Considerations

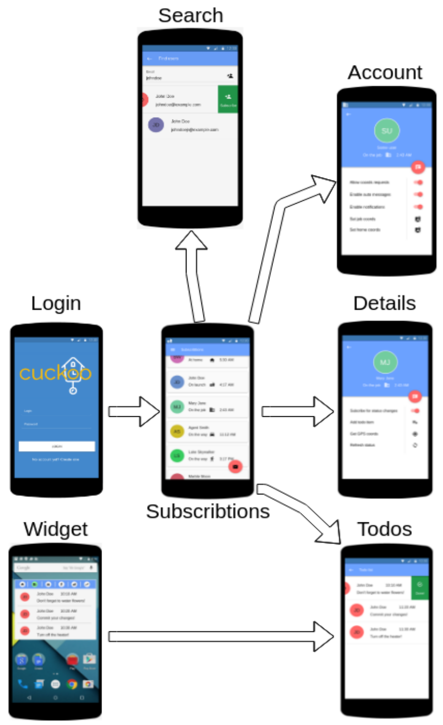
How will your app handle data persistence?

App will store all data in Firebase Realtime Database, because all authorized users can edit each other data.

App will store a copy of user’s todo list in local DB to make app able to show todo list in offline mode. When device is offline, al data stores in local DB and then sync with Firebase Realtime Database when device is online.

Describe any edge or corner cases in the UX.

Here is a scheme of user navigation flow



Describe any libraries you’ll be using and share your reasoning for including them.

Lib/tool/other	Version
Android Studio	3.1.3
com.android.tools.build:gradle	3.1.3
Gradle	4.7
Picasso	2.71828
Firebase Realtime Database	16.0.1
Firebase Notification	17.0.0
Firebase Messages	17.0.0
Firebase Auth	16.0.2
GooglePlay Services (play-services-places, play-services-location, play-services-maps)	com.google.android.gms:play-services-maps:15.0.1 com.google.android.gms:play-services-places:15.0.1 com.google.android.gms:play-services-location:15.0.1
hdodenhof/CircleImageView	2.2.0
Android Support Library	v7:27.1.1
LiveData and ViewModel	1.1.1
Room for local DB persistance	1.1.1
RxJava2	2.1.10
RxAndroid	2.0.2
Gson	2.3.0
Retrofit	2.4.0
Moxy	1.5.3

Dagger2	2.14.1
Android Data Binding	-
Espresso	3.0.1
JUnit	4.12

Describe how you will implement Google Play Services or other external services.

For this app I will implement an Firebase Realtime Database, Firebase Auth, Firebase Notifications and Messaging.

For in app notifications I'll add Firebase Cloud Functions, it will be triggered when:

- New todo item added
- User marked todo imem as "Done"
- User changed his status

Next Steps: Required Tasks

Techical details

- Minimum Android API – 16.
- App is written solely in the Java Programming Language.
- App keeps all strings in a **strings.xml** file and enables RTL layout switching on all layouts.
- App uses IntnetService to communicate with widget.
- App uses LiveData and ViewModel to make DB calls easisy.

Task 1: Project Setup

1. Init deafult project:
   - Minimal Android API – 16
2. Add gradle dependencies
3. Add Manifest permissions
4. Create app structure skeleton:
   - ui – for Activities, Fragments, Presenters
   - model – for Room entities and DAO objects
   - di – for Dagger components
   - services – for service code
   - infrastructure – for app internal utils and helpers
5. Implement basic Dagger classes, inject Retrofit instance

Task 2: Implement app models

1. Create “User” model
2. Create “TodoItem” model
3. Create “Subscribtion” model
4. Create “Message” model
5. Create local DB
6. Create DB context and DAO objects for every model
7. Implement models saving into local DB

Task 3: Implement UI for Each Activity and Fragment

1. Build UI for Login activity
2. Build UI for Login Subscribtions activity
3. Build UI for slider menu
4. Build UI for Login user detail activity
5. Build UI for Login Account activity
6. Build UI for Login Search activity
7. Build UI for Login Todos activity
8. Build UI for app widget

Task 4: Implement internal activity logics

1. Implement navigation for activities
2. Create dummy model instances and dummy event listeners
3. Implement models passing through the activities
4. Implement RecyclerView adapters
5. Save user settings into SharedPreferences
6. Implement IntnetService to communicate with app widget
7. Implement JobDispatcher to update user data at regular interval
8. Implement databinding
9. Add final event listeners

Task 5: Implement geofences logics

1. Create Geofences service
2. Implement home/work coords setting from Google Maps

3. Change user status when geofence triggered

Task 6: Implement Firebase app integration

1. Create Firebase account, add google-services.json into project
2. Create Firebase Realtime Database
3. Implement Firebase Auth
4. Show notifications, created from Firebase console (manually)
5. Remove dummy models

Task 7: Implement database triggers

1. Implement Firebase Cloud Functions for next triggers:
    - New todo item added
    - User marked todo imem as “Done”
    - User changed his status