Architectural Design

Overview

Our overall architecture for the ‘Emergency Button’ (working title) system is a client-server based system. Carers, operators and carees (clients), will connect to a central server. All communication between clients will pass through the server, and keep track of the current state of affairs.

We will be utilising Parse for our server hosting, and the Parse API to handle client-server communication. We opted for Parse because they offered a ready to develop suite of tools for Android, our current target platform. Additionally, should we desire to expand to other platforms, Parse also offers SDKs for Windows Phone and iOS.

The server will also hold a database storing important information such

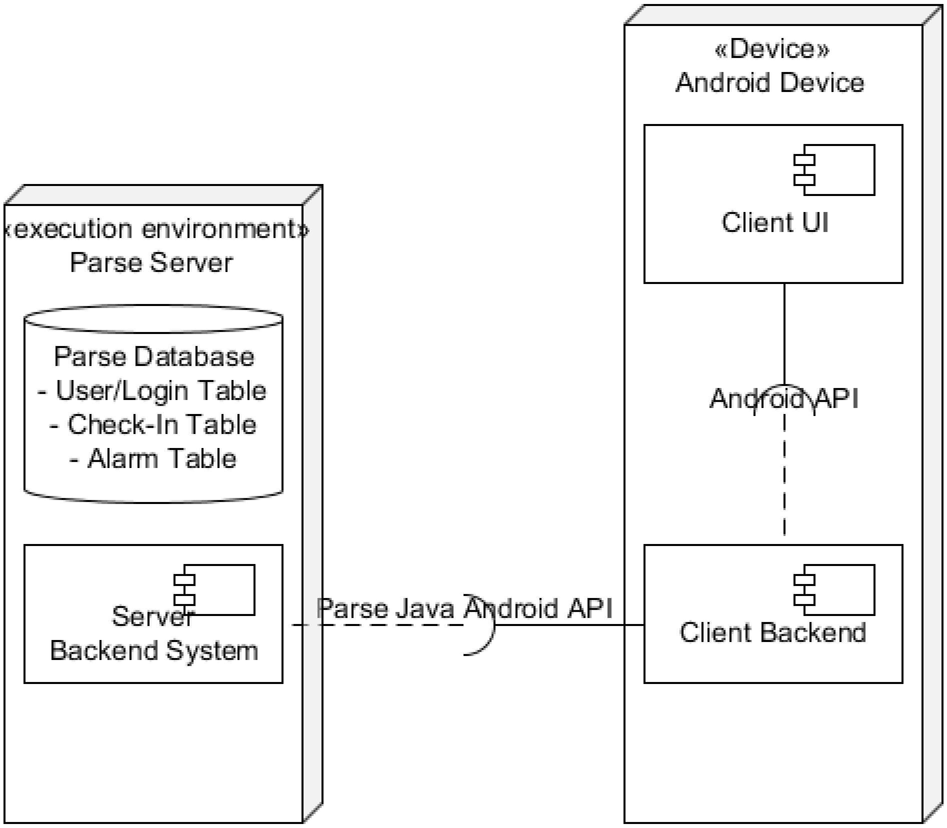
The server will keep track of information such as ‘last check-ins’ and ‘active alerts’, and also listen for messages from clients, such as check-in acknowledgements, and alerts. The server will handle any pushing of such notifications.

Another component is the Client UI. Client UI presentation will be hired client side. In the case of the carers and carees, this will be handled by the device itself. Operators will utilise a desktop application to keep track of information. The Client UI will interface with the Client Backend System, and therefore the Server Backend System.

Definitions:

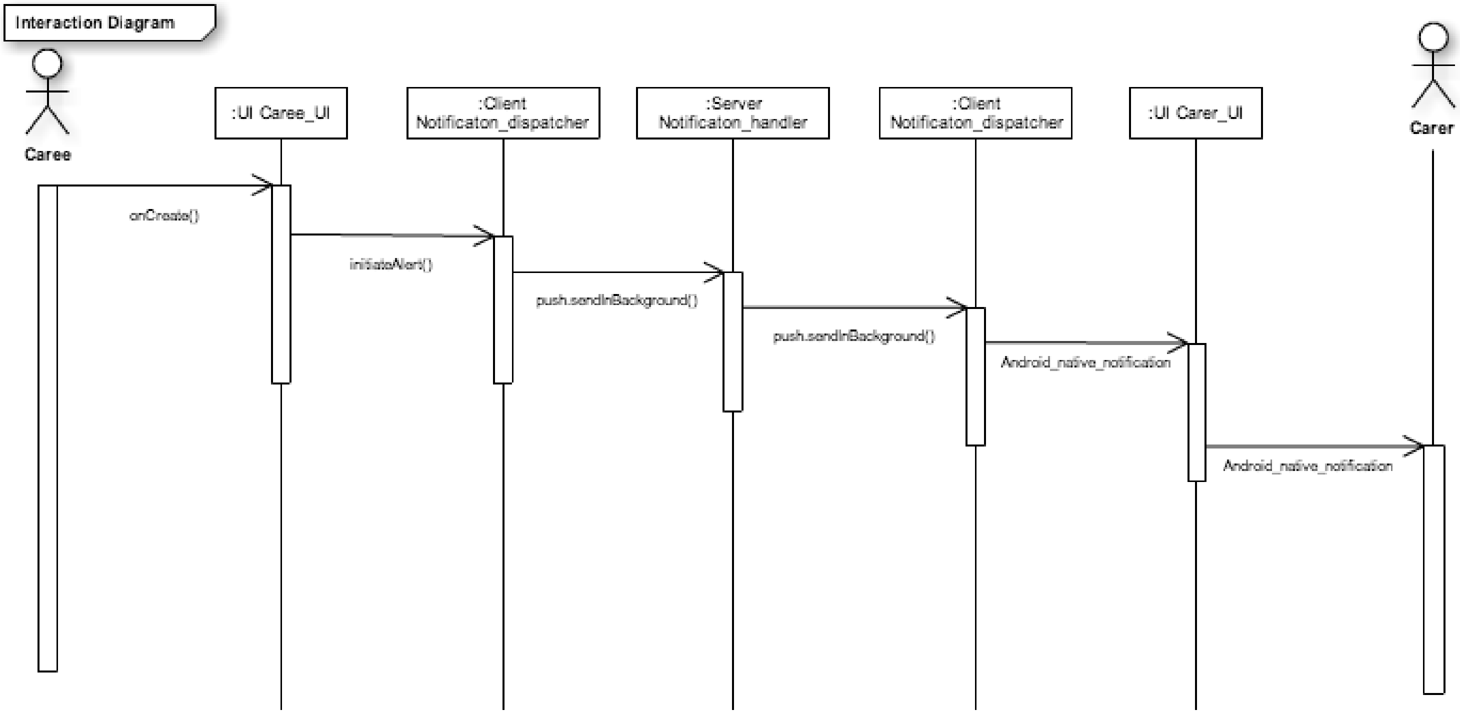
Client: Any user who interfaces with the central server. Includes carers, carees and operators.
Client UI: The compeonent of the client that acts as the interface of interaction (eg. to send message, alerts) and also feedback (eg. to display messages, alerts).
Client Backend System: The component of the client that handles interactions with the server through the Parse API.
Server: The internet-facing system server running on Parse.
Server Backend System: The component of the server that handles interactions with the client. It makes use of the Parse push system.

Structure

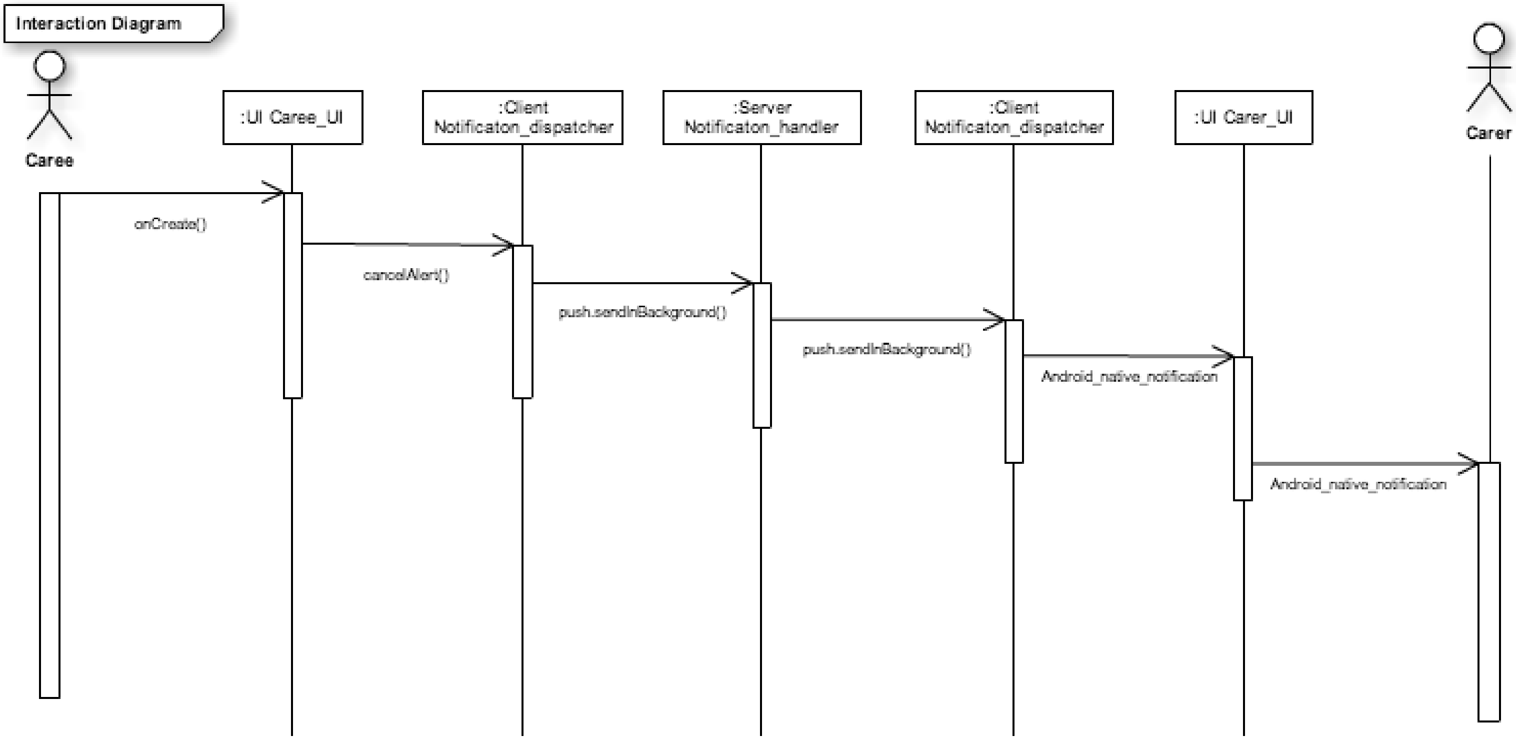

«execution environment»
Parse Server
Parse Database
- User/Login Table
- Check-In Table
- Alarm Table
Server
Backend System
Parse Java Android API
«Device»
Android Device
Client UI
Android API
Client Backend

Interactions

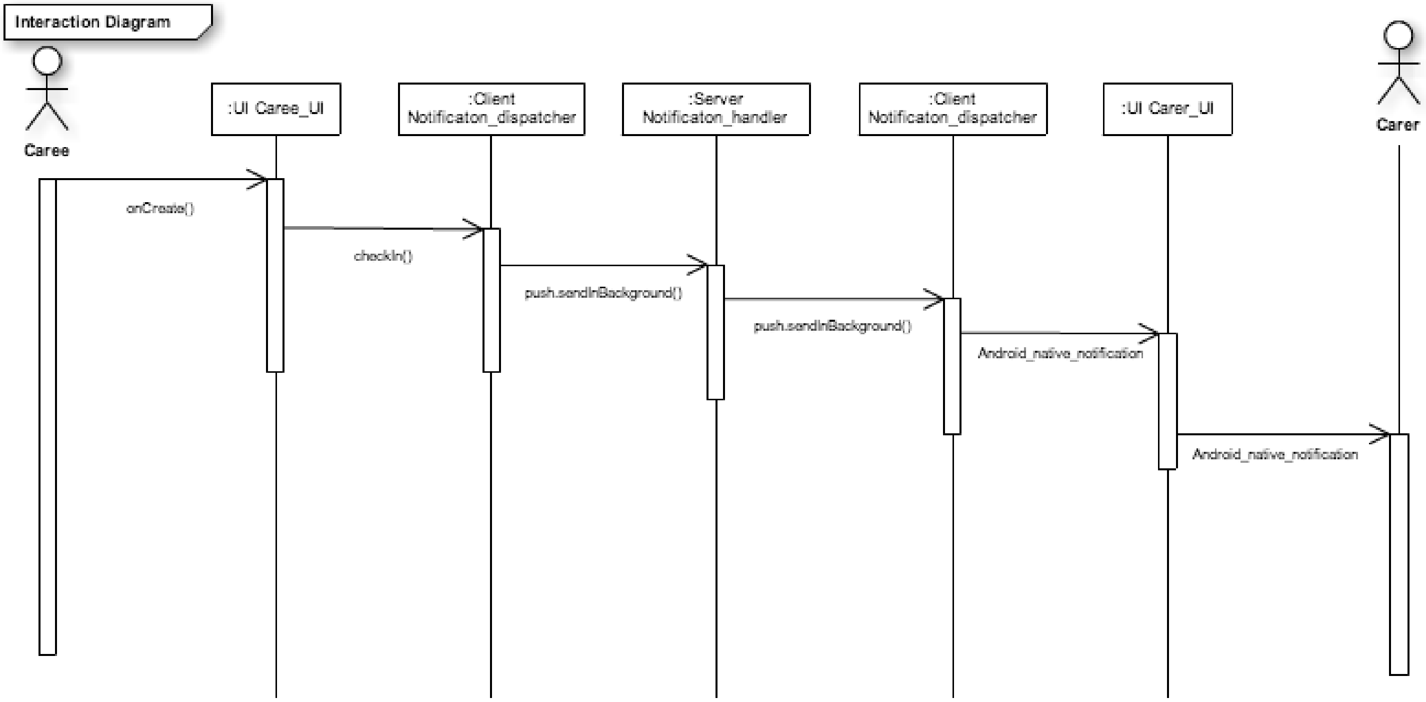
Feature 1 - Patient should be able to send alert
Feature 3: Carer should receive an alert



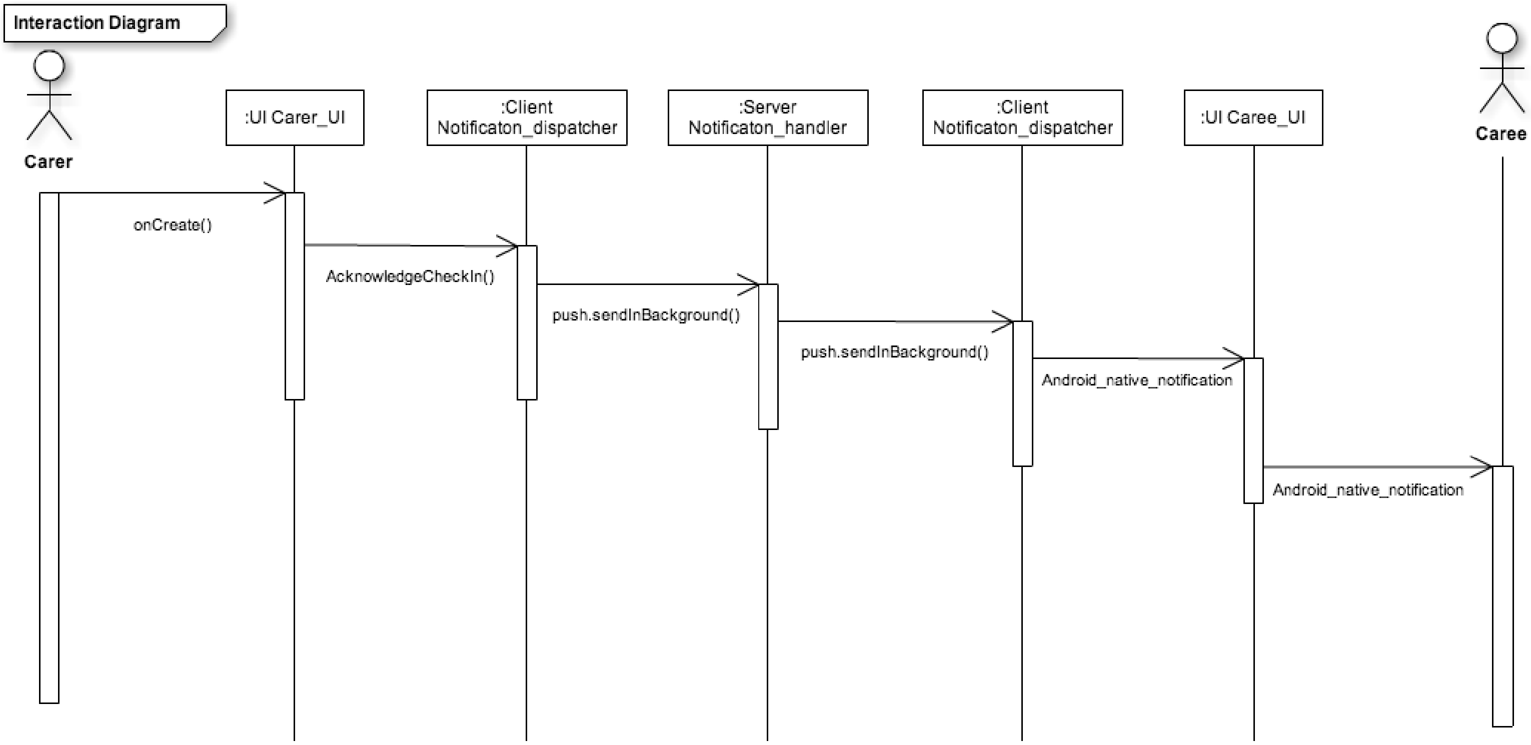
Feature 2: Patient can cancel an active alert



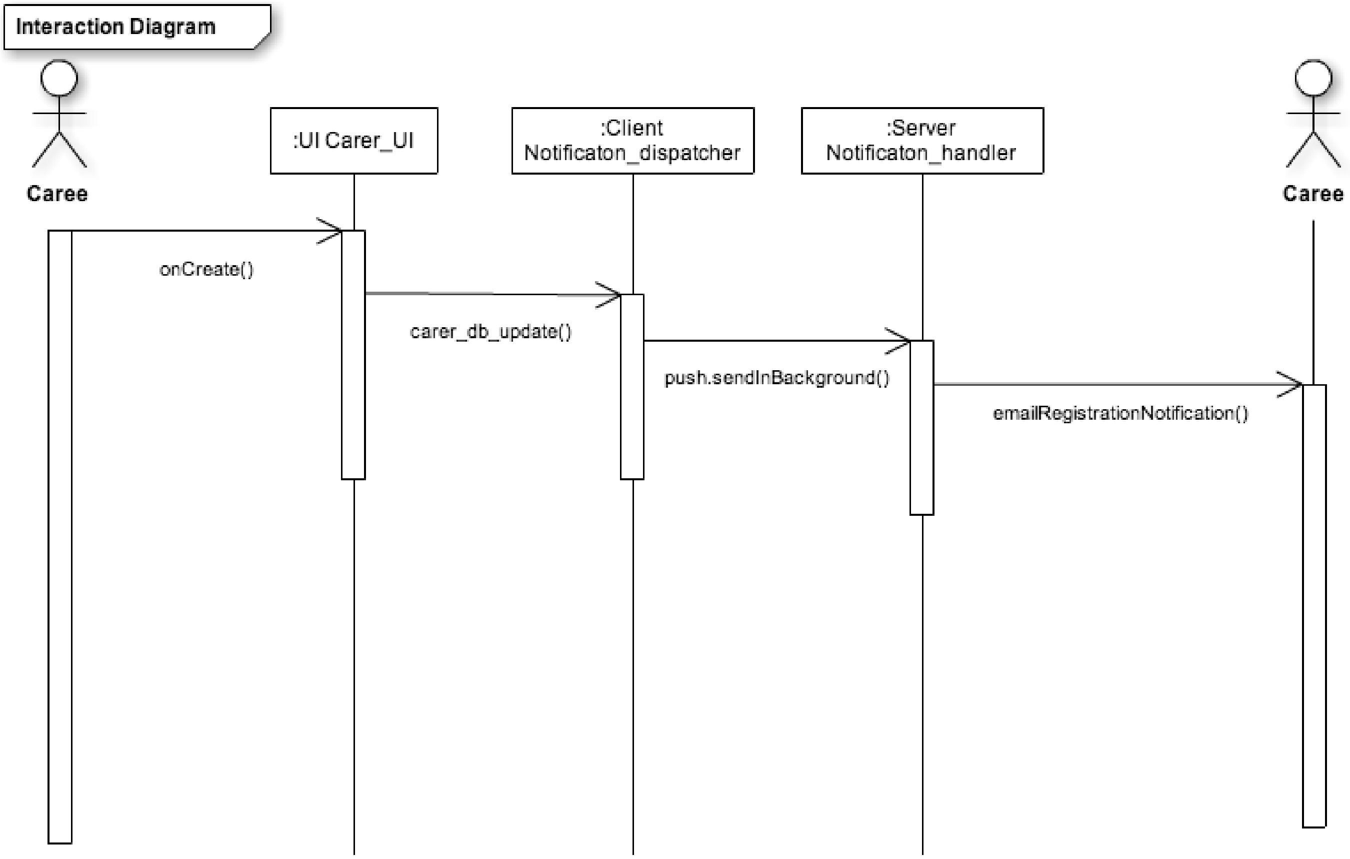
Feature 4: Patient should be able to regularly 'check-in' to confirm they are in good health
Feature 5: The carer should receive a notification that their patient has checked-in


Interaction Diagram
Caree
:UI Caree_UI
:Client Notificaton_dispatcher
:Server Notificaton_handler
:Client Notificaton_dispatcher
:UI Carer_UI
Carer
onCreate()
checkIn()
push.sendInBackground()
push.sendInBackground()
Android_native_notification
Android_native_notification

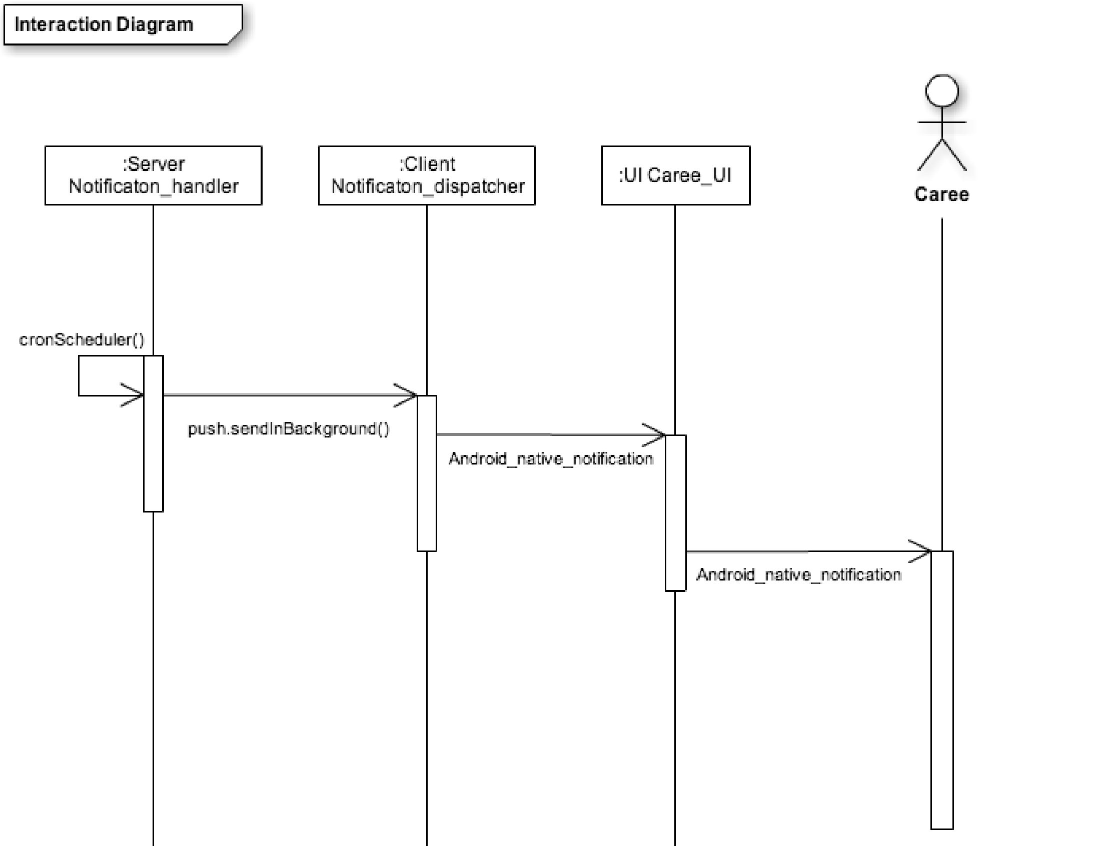
Feature 6: The carer should be able to acknowledge a patient's check in.


Interaction Diagram
Carer
:UI Carer_UI
:Client Notificaton_dispatcher
:Server Notificaton_handler
:Client Notificaton_dispatcher
:UI Caree_UI
Caree
onCreate()
AcknowledgeCheckIn()
push.sendInBackground()
push.sendInBackground()
Android_native_notification
Android_native_notification

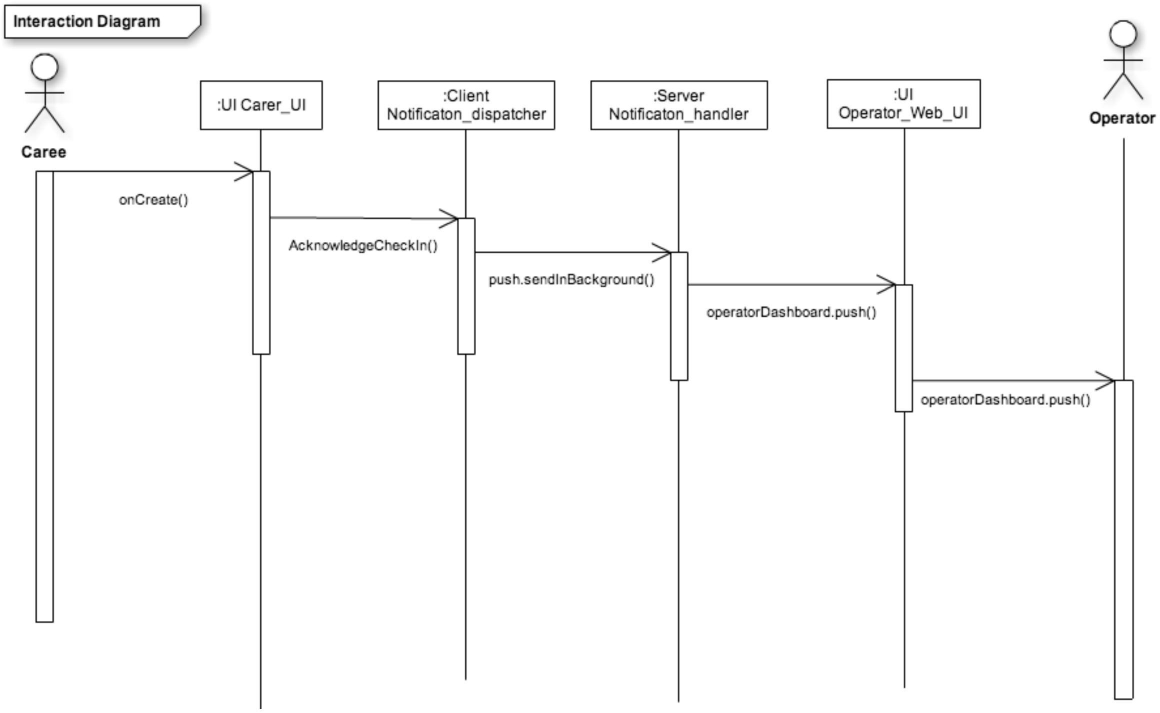
Feature 7: The patient should be able to change their list of carers at will.



Feature 8: The carer should receive a notification that their caree has failed to check-in


Interaction Diagram
:Server Notificaton_handler
:Client Notificaton_dispatcher
:UI Caree_UI
Caree
cronScheduler()
push.sendInBackground()
Android_native_notification
Android_native_notification

Feature 10: Operators are notified if no carers respond to a help request from a caree



Interfaces

Client Backend System

The interface to push notifications (to call for alarms, check-in, etc.) is performed through the extremely convenient Parse API.

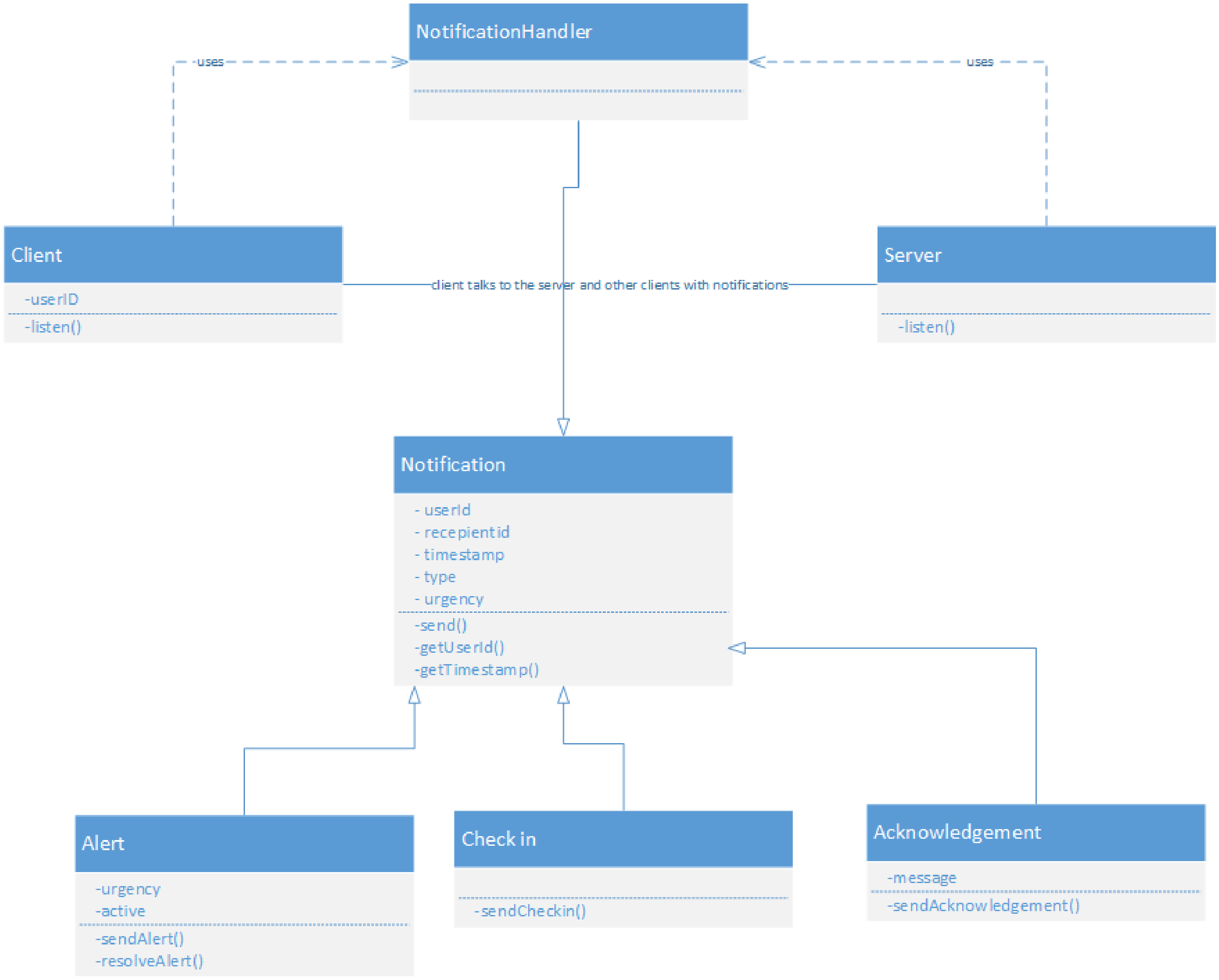
```
// Parse API requires a setup in order to begin pushes
PushService.setDefaultPushCallback(this, some.class);
ParseInstallation.getCurrentInstallation().saveInBackground();

// Sample code to send an alert using the Parse API
public void initiateAlert(View view){

        // Code that Inserts/Logs into Database
        ParseObject testObject = new ParseObject("Alarm");
        testObject.put("Name", "Harlan Wade");
        testObject.put("Activated", true);
        testObject.saveInBackground();

        // Code that Pushes
        ParsePush push = new ParsePush();
        push.setChannel("Carers");
        push.setMessage("Harlan Wade needs HELP!!!");
        push.sendInBackground();
```

A basic class diagram for our first thread of functionality. The central notification class is extended to allow for all sorts of messages.



Parse Operator Interface

We will use a web-based client for the operator interface. This will be hosted on Parse’s ‘Cloud Code’.

We will utilise the ‘express’ framework since parse comes with it. This will also allows us to use the standard parse objects such as Parse.User