Health Alert
Technical Documentation

v2.0
December 7, 2014

CONTENTS

1. Introduction

1.1 What is Health Alert?

Health Alert is a mobile application developed in the fall of 2014 to fulfill the requirements of Duke University's Computer Science 408S course by a team of undergraduates. Our team worked with Dr. Ryan Shaw of the Duke University School of Nursing over the course of this project.

This application provides a health monitoring solution for doctors, patients, and caretakers. It connects with third-party monitoring devices and processes the data via Apple's HealthKit API (Application Programming Interface). If certain health metrics are outside of ranges predefined by physicians, we send an alert to the patient's doctors and caretakers that something may be wrong. The goal of this application is to provide these caretakers and doctors with real-time reporting about a patient's health symptoms and to make sure that these potential health issues are dealt with immediately.

An important disclaimer is that this application is not intended to be a replacement for traditional emergency reporting mechanisms such as 9-1-1, and it is instead intended to be a supplement to the preexisting set of tools available to patients, caretakers, and physicians.

1.2 Who is this guide for?

This guide is intended to make it easier for technical developers to extend and customize the Health Alert codebase to suit the needs of their audience. For example, if a department within a hospital wishes to trial this application, we provide instructions on how to provision and install this application on their proprietary databases, and allows them to extend and modify the functionality to suit their needs.

End users of the application, including doctors, patients, and caretakers, will find it much easier to read our User Guide, which will provide simple instructions on how to get up and running with Health Alert.

1.3 Legal information and licensing

Our application is open source and governed under the GNU General Public License, version 3, published on June 29, 2007. Full information about this license can be found in our project files under `/doc/LICENSE.txt`.

We believe that this application has the potential to benefit a wide range of end users, which is why we decided to make it open source and modifiable by others wishing to use our software. Our team actively encourages others to find new, cool use cases for Health Alert, especially those that we did not even think of ourselves.

We do ask, however, that proper attribution is maintained on each modified copy of source code, citing Wei Chen, Ethan Gottlieb, and Chinmay Patwardhan as the original developers of the application, and Dr. Ryan Shaw as the original client.

2. Installation

2.1 Installing the source code

Our source code is maintained in a public repository. It can be accessed via the URL https://github.com/duke-compsci408-fall2014/PropelPro. Once on the webpage above, anyone can view the files related to our project.

In order to download the source code of the application onto your machine, follow the following easy steps. For the example below, we are going to store all of the code in a folder called `HealthAlert` which lives within our `Documents` folder.

You must launch the Terminal application on your machine in order to get the magic steps below to work correctly.

Step 1. Create the HealthAlert folder and navigate to it:

```
Ethans-MacBook-Pro:~ ethangottlieb$ cd Documents
Ethans-MacBook-Pro:Documents ethangottlieb$ mkdir HealthAlert
Ethans-MacBook-Pro:Documents ethangottlieb$ cd HealthAlert
```

Step 2. Download the project and its dependencies into your HealthAlert folder:

```
Ethans-MacBook-Pro:HealthAlert ethangottlieb$ git clone
https://github.com/duke-compsci408-fall2014/PropelPro.git
```

If you've done the above steps correctly, you should be seeing the following text at the bottom of your page:

```
Cloning into 'PropelPro'...
remote: Counting objects: 1039, done.
remote: Compressing objects: 100% (579/579), done.
remote: Total 1039 (delta 527), reused 875 (delta 438)
Receiving objects: 100% (1039/1039), 2.63 MiB | 0 bytes/s, done.
Resolving deltas: 100% (527/527), done.
Checking connectivity... done.
```

If you don't, you can troubleshoot issues via GitHub at the following link: https://help.github.com.

2.2 Installing XCode

We developed our application on Apple's XCode software, which requires a Mac computer to work. Using XCode, you can develop the app, as well as emulating it on your local machine.

You can install the latest version of XCode by accessing the following link: `https://developer.apple.com/xcode/downloads/`.

Once you get that up and running, open up the file `/HospitalTabbedApp.xcodeproj` and it will launch on XCode. You are now able to edit your file on XCode!

3. Dependencies

3.1 Introduction

The Health Alert application relies on a variety of third party applications in order to function properly. In this section, we go over the parts of the application that rely on third parties to function properly. This section does <u>not</u> cover third party databases; this is instead described in the next section.

3.2 Notifications with Twilio

Health Alert leverages the Twilio API for the messaging and calling functionalities of the app. This component runs on a separate Heroku server. The guts of this application can be found in the folder `/Twilio Code`, but we do not dive into the specifics of that code here – that code is built to run on any server.

Instead, navigate to /Hospital Tabbed App/Constants.swift, where we are able to change the configuration of the database files. For Twilio, to the following code:

```
// Base URLs for Twilio API:
private static let URL_TWILIO_SERVER = "http://dukecs408-
   twilio.herokuapp.com"
static let URL_TWILIO_CALLS = "\(URL_TWILIO_SERVER)/notifyWithCall?"
static let URL_TWILIO_TEXTS = "\(URL_TWILIO_SERVER)/notifyWithText?"
```

Here, you can see that we have two separate notification calls – `URL_TWILIO_CALLS` which is the URL for the server endpoint where the calling functionality is located, and `URL_TWILIO_TEXT`, the endpoint where the texting functionality is located. This is built into the app.

If you want to start up a new server instance, we have made it easy for you to simply change the location of the server. Simply change the URL of the `URL_TWILIO_SERVER` field to the server location where you wish to install the application. Our application takes care of the rest.

To develop further with the Twilio API, please visit their site at:
`http://www.twilio.com/docs` for complete documentation.

4. Databases

4.1 Our server structure

As mentioned in section 4, we maintain two separate databases for our application. The first is the Twilio server, which is responsible for all text and call notifications. The second part of our server is responsible for all updates, additions, deletions, and reads from the external database. This currently is hosted via the Duke Co-Lab's Bitnami server. We have made it easy, however, for you to change a few lines of code to deploy to a different server.

Our server itself is divided up into the following tables:

- Bounds – this is where we store information on the bounds that would require a notification.
- Contacts – this is where we store all information for contacts.
- Doctors – this is where all information for doctors is stored.
- Notifications – this is where information on the types of notifications our contacts want to receive is located.
- Patients – this is the table where all information about patients (end users of the application) is stored.s

4.2 Where to change fields

We are using the Bitnami server in order to host the HealthAlert application. We have made this easily extendible to host on whichever server you want, as long as the server is a MySQL-based server (all of our calls are written in MySQL). If you wish to change to a different database query language, please refer to our Technical Documentation document for information how to do that.

4.3 Changing server endpoints

There are two main places to change server endpoints in the application. First navigate to `/PropelPro/Database/ConnectionInfo.php`. There, you should see the fields `$username` and `$password`. These must be changed in accordance with your new database's username and password.

Next, navigate to `/Hospital Tabbed App/Constants.swift`. This application contains all endpoints for `INSERT`, `SELECT`, `UPDATE`, and `DELETE` functions on your database. Simple switch these URLs if your database calls happen at different locations. We have organized these URLs to correspond to `bounds`, `contacts`, `doctors`, `notifications`, and `patients`.

One important field is `private static var URL_DB_SERVER`. This is the endpoint of the URL of the server. Make sure to update this with the correct URL of your server

5. Technical Tour: How to Extend Health Alert

5.1 Introduction

This section is intended to give you a brief tour through the technologies powering the Health Alert project. We try to be as comprehensive as possible without diving into the specifics of the implementations of the dependent technologies. For these, we provide links to the documentation written by the original developers of these applications (the people that know them best).

We also give a brief overview on the relevant files in our XCode project without going into details of the Swift programming language. We have provided a bunch of documentation for Swift that we found helpful in the development of our application. However, when we first wrote Health Alert, Swift was a mere two months old and without a lot of great documentation on the internet. We hope that by the time you are developing with Health Alert, the Swift documentation is a lot more robust.

5.2 iOS development with Swift: the guts of Health Alert

Most of the functionality of Health Alert is built in Swift. Here we go through each file that is important to the development of the app. These files can be found in `/PropelPro/Hospital Tabbed App`:

- `AddContactController.swift` – the view controller that connects to the add contact functionality in the application.

- `AddPhysicianController.swift` – the view controller that connects to the add physician functionality in the application.

- `Constants.swift` – contains information for all of the constants in the application, including database endpoints and the URL for the Twilio server.

- `Contact.swift` – specifies the model for the contact object that will be stored in a table in the main database server.

- `ContactsVC.swift` – the view controller that connects to the add contacts functionality in the application.

- `Doctor.swift` – specifies the model for the doctor object that will be stored in a table in the main database server.

- `DoctorsVC.swift` – the view controller that connects to the add/remove doctors functionality int eh application.

- `EditContactController.swift` – view controller that connects to edit contact information in the application.

- `EditDoctorVC.swift` – view controller that connects to the edit doctor functionality in the application.

- `GlobalHealthStore.swift` – interacts with the Health Kit API to bring in information.

- `HomeVC.swift` – view controller for the home page of the app.

5.3 Developing with Twilio

The code that runs the Twilio application is found separately within the app. It is written in node.js and can be found in the folder `/Twilio Code` or at the URL `https://github.com/chinnychin19/healthAlertTwimlServer`.

The important parts of this code are located in the following areas:

- `app.js` – contains all of the application code that enables notifications via call or text. It is written in Javascript. For more information on how to develop in nods.js or Javascript, visit `http://nodejs.org/api/`.

- `Package.json` – initial code that needs to be run.

For information on deploying to the Heroku server, visit the Application Maintenance Documentation.