MedIT

Release Notes
Version 1.0 - Clinic Workflow Optimization Mobile App

1 Overview

The purpose of this application is to improve the workflow optimization of clinics associated with Children's Healthcare of Atlanta. Dr. Donald Batisky, a pediatric physician at Children's Healthcare of Atlanta, served as our client for this project. Dr. Batisky observed that the stem of his clinic's inefficiencies resulted from patients not arriving to their appointments on time.

Fundamentally, the issue derives from a lack of communication between the patients and the clinic. Patients fail to notify the clinic beforehand if they do not plan on going to the appointment. In some cases, patients do not have enough information to reach the clinic on time. They may be unaware of traffic or weather delays or simply underestimate the time it takes to get to the clinic. This lack of information results in prolonged wait times in the clinics and increased frustration amongst medical staff and patients.

To address these issues, we created a patient-facing mobile application called MedIT. This application is developed on the Ionic platform, making it available on most modern mobile operating systems, including iOS and Android.

The proposed functionality of our app revolves around the following tasks:

- Allow patients to view, confirm, and receive notifications about upcoming appointments
- Remind patients of upcoming appointments
- Allow patients to cancel appointments
- Receive traffic and weather updates that may affect travel time to appointments
- Concise information about navigation to the clinic and parking at the facility
- Allow returning patients to inform clinic about unchanged application information (e.g. contact information, home address, insurance provider)

1.1 Scope

Given the purpose of the application and the proposed features, it is important to list the scope of the mobile application that has been developed. This application is a **front-end plugin** to the the data stream about patient appointment information being received from the Epic EHR. During our development, we were unable to connect to the Epic EHR, thus requiring us to use a mock database of patient information.

Connecting to the Epic EHR is possible, but it requires working with the Epic customer support from the hospital's server end. Due to the limited time available for this project, implementing this step was not possible. However, we have developed this Ionic application keeping in mind the notation patient appointment information may be in from the Epic EHR. Therefore, when a secure connection is established with the Epic EHR to extract patient appointment information, our application can serve as a front-end plugin for patients to use.

This application also **does not support account creation**. For future development, the application would depend on patients first registering on MyChart (https://www.mychartcentral.com/Home.aspx), which will give patients a login to the Epic EHR to view limited information about themselves. They would then go on to use these MyChart credentials to login into our application and connect themselves with their appointment information.

2 Release Notes

2.1 Installation Instruction

We are first assuming that latest version of Java is already installed.

The minimum system requirement to run this project is to have Windows 7 or later. We will be installing the following software: NodeJS, Cordova, Ionic, MySQL, and Maven

1. The first step will be to install Node.js. Follow instructions to download and install the latest version of Node.js for your Windows machine on the following link: https://nodejs.org/en/download/

2. Open the command prompt (cmd) application. We need to install the latest version of Cordova Apache. Type the following command to install it:
   ```
   $ npm install -g cordova
   ```

3. If you would like to develop on the Android platform, you may need to install the platform-specific tools. Instructions can be found here:

   http://cordova.apache.org/docs/en/3.4.0/guide/platforms/android/index.html#Android%20Platform%20Guide

   This step is not necessary for the subsequent steps, so if you are unsure, you may always come back to this step later.

4. Extract and navigate to the application directory. Run the following command to build the dependencies of the project:
   ```
   $ npm install -g ionic
   ```

5. Follow the MySQL installation instructions at http://dev.mysql.com/doc/refman/5.7/en/windows-installation.html
   During installation, you should create a MySQL user with the following attributes
   a. Username: serverUser
   b. Password: gtsecret
   You are welcome to change these to give more secure login credential to your MySQL installation, but you should make sure to change the appropriate variable definitions in DatabaseManager.java in the Server source code.

6. Follow the Maven installation instructions at https://maven.apache.org/install.html

7. You are now ready to run, test, and develop this application.

2.2 External Services

This project uses a few external services to function. This section includes a short description of each, including login credentials used for the development of this project.

2.2.1 Twilio

Twilio is a service which provides access to various communication technologies through an online API. This project uses Twilio to send SMS reminder messages. More about Twilio can be found at https://www.twilio.com.

A personal account was used to develop with Twilio. For deployment, a new account should be registered at https://www.twilio.com/try-twilio. Once you've created an account, go to https://www.twilio.com/user/account and press "Show API Credentials" to get the credentials you will need to copy into the Server source code.

2.2.2 Gmail

Gmail is a Google email client. This project uses Gmail to send email reminder messages. More about gmail can be found at https://www.gmail.com.

A gmail account was created for the development of this app. You can access this account with the following credentials:

Username: medit.auto
Password: gtsecret

2.3 User Manual / Design and Test

Once all the required software is installed, the project can now run. We have three different parts to our project that must concurrently run in order to test the full potential of our project. The three parts are the application, the server, and the database. In order for the application to run smoothly, both the server and the database should run first.

2.3.1 Setting Up the Database

Take the following steps to setup the medIT MySQL Database.

1. Make sure the MySQL service is running. For most installations, MySQL will start automatically when Windows launches. If you selected the manual start option during installation, you will need to start the service ‘MySQL57’ in the Windows Service Manager. You can access the service manager by searching for and running “Services” in the start menu.

2. Run the MySQL Command Line Client. You will need to enter the password you used when installing MySQL.

3. Execute the following command to create a database called “medit”
   ```
   > create database medit;
   ```

3. Execute the following command to switch to the new database
   ```
   > use medit
   ```

4. Execute the following command to load the medIT schema into the database
   ```
   > source <medIT directory>\DB\create_tables.sql
   ```

5. Execute the following command to load mock data into the database
   ```
   > source <medIT directory>\DB\populate_data.sql
   ```

6. The database is now initialized. You can now run the server.

2.3.2 Building and Running the Server

Before running the server, make sure you have copied the necessary external service credentials for Twilio and Gmail. The locations these credentials need to be placed are marked with a TODO comment. More about these external services can be found in section 2.2.

Take the following steps to build and run the medIT Server:

1. Open the Command Prompt.

2. Navigate to the ‘Server’ directory.
   ```
   $ cd <medIT directory>\Server
   ```

3. Build the project with Maven
   ```
   $ mvn clean package
   ```

4. This will produce, among other artifacts, two *.jar files in the newly-created ‘target’ directory, within the ‘Server’ directory: medIT-1.0-SNAPSHOT.jar and medIT-1.0-SNAPSHOT-jar-with-dependencies.jar.
   Maven will handle downloading and managing the dependencies needed to build and run the Server. For reference, the Server depends on the following external libraries:
   a. Twilio - a service for sending and receiving SMS messages
   b. org.json - a library for parsing and building JSON data
   c. javax.mail - a library for sending emails
   d. mysql-connector-java - a library for interacting with MySQL databases from Java

5. You can now run the Server with the following command (still from the ‘Server’ directory):
   ```
   $ java -jar target/medIT-1.0-SNAPSHOT-jar-with-dependencies.jar
   ```

   Note: The MySQL service must be running before the Server is run.

2.3.3 Running the Ionic Application

Take the following steps to run the Ionic application locally in a web browser:

1. Open a command prompt at the directory “demo”

2. Then start up ionic by running the following line.
   ```
   $ ionic serve --lab
   ```

3. The command prompt might ask you to choose which server to use. Currently, we only have our application available with localhost server. Enter the number for the localhost.

4. This will open the application with user’s default browser and will the user to the login page.

5. Please enter ***atsou3*** or ***avijay3*** or ***skim3*** or ***pbothra3*** or ***mbarulic3*** as username and ***password*** as password. This application requires MyChart credentials, but since we can’t use any existing credentials nor do we have access to it because of the limitation of the server, the username/password that will be used is a dummy account.

6. In order to debug the application the following flags can be used when deploying the server.

   ```
   $ ionic serve --lab -c -s
   ```

   The ‘-c’ flag activates console logs whilst the ‘-s’ flag activates server logs. More information regarding debugging applications can be found at: http://ionicframework.com/docs/v2/resources/developer-tips/

2.3.4 Testing the Ionic Application Natively

In order to test out the Ionic application on native devices the following procedure is required:

1. First, create an account with Ionic using the following webform https://apps.ionic.io/signup

2. Once created, download the 'Ionic View' application to the desired phone on which you would like to test the application from the following website: http://view.ionic.io/

   Note: The 'Ionic View' application needs to be downloaded on every device that you would like to test the application.

3. Once downloaded, using the instructions detailed in the following manual to upload the Ionic app to your account and test it on a native device using the 'Ionic View' application. http://ionicframework.com/docs/cli/uploading_viewing.html

Note: Until the server is hosted on a third party service, the application will not be able to authenticate any users and hence testing for all features will not be possible.

2.4 Source Code

The source code for this project can be found at:
https://github.com/vanugrah/MedIT

3 Moving Forward

The Ionic Application and Java server are able to work together since both of them follow strict rules on data notation (JSON). If, however, you would like to modify the notation of the objects being handled, you will need to modify the source code for both the Java server files as well as the Ionic Application.

The current notation of our appointment object is as follows:

```
CREATE TABLE Appointment (
        AppointmentID int NOT NULL AUTO_INCREMENT,
        PatientID int NOT NULL,
        Username varchar(255) NOT NULL,
        DoctorID int NOT NULL,
        ClinicID int NOT NULL,
        Date_of_appt datetime NOT NULL,
    Confirmed bit(1) NOT NULL,
    Checked_In bit(1) NOT NULL,
    Cancelled bit(1) NOT NULL,
        Notes text NULL,
  Date_of_last_reminder date,
        PRIMARY KEY (AppointmentID),
        FOREIGN KEY (PatientID)
                REFERENCES Patient(PatientID)
                ON UPDATE CASCADE
                ON DELETE CASCADE,
        FOREIGN KEY (Username)
                REFERENCES Parent(Username)
                ON UPDATE CASCADE
                ON DELETE CASCADE,
        FOREIGN KEY (DoctorID)
                REFERENCES Doctor(DoctorID)
                ON UPDATE CASCADE
                ON DELETE CASCADE,
        FOREIGN KEY (ClinicID)
                REFERENCES Clinic(ClinicID)
                ON UPDATE CASCADE
                ON DELETE CASCADE
) ENGINE = InnoDB;
```

The application cannot be deployed to a portable device (i.e. cell phone, tablet) yet because of the limitation of the localhost server. In order to deploy the application with working functionality, the host server must be from a third party.

Our proposed functionality included many features, but unfortunately, not all features were completed. The following is a list of features that would complement the overall goal of the application or incomplete features from our initial list of proposed features:

- Traffic API (https://developers.google.com/maps/documentation/directions/)
- Weather API (https://www.wunderground.com/weather/api)
- Parking Instructions
- Push Notifications
- Calling clinic through the app

Once your application has been finalized, it can be published on both the Google Play Store as well as the Apple Store. Instructions for publishing can be found at the following link: http://ionicframework.com/docs/guide/publishing.html

Keep in mind that accounts must be created in each operating system's store (Google Play, App Store). These may involve one-time or recurring fees. As of the date this release note was released, the fees for creating accounts on popular operating system's stores are as follows:

- Android: $25 (one-time fee)
- iOS: $99 (recurring annual fee)