Team SeniorPanda: Steve Siyang Wang, Justin Zhang, Sanmay Jain
Client: Ray Zwycewicz
AppName: SeniorPanda (PD101)
Date Last Updated: Dec 14, 2014

Technical Documentation

Overall Architecture and Design

For the overall architecture of the app, AndroidManifesto.xml governs the layout of the UIs, the interactions and relative layouts between them. In the project, each activity class represents a panel/ UI in the android app.

The overall structure of app is elaborated in the following walk-through of the app, which presents how initial login, authentication, and medication record is set up.

When a new user first opens the app, the app checks with the database to validate the user. This is done by MainActivity.java, LoginActivity.java, in conjunction with Backend.java class which helps facilitates interaction with the backend. When the MainActivity.java is created, the onCreate() gets invoked, and the tryInitMainScreen() will be invoked. It check to see if the authentication, and medication record is set. If yes, it jumps to unlock screen; whereas if not, it will jump to authentication / login page, and medication record setup page. The details of the two process is elaborated below. The authentication is done by interacting with DreamFactory (provided by Bitnami, which wrapps around the mySQL running on EC2 virtual machine), whose task is then delagated to the DreamFactoryGetJob.java and other two class in the job packet. The models package, which contains models like UserDataModel.java, LoginModel.java is used to represent the Model View Control structure so that data and views can be incapusulated, which is also an encouraged design pattern. The classes in the models package interact with the DreamFactory JSON objects to receive and break down information from the database. The fromJson method in the SymptomDataModel.java class is an example of this functionality. They also help package information entered by the user into JSON objects to send to the database. The toJson method written in the SymptomDataModel class does exactly that.

After the user is validated via the database, the LockActivity.java comes into place, and prompts usre to set a pin as well as the user name. Here the username and pin will be stored in the PrefUtil file, which is like a storage file in Android that will persist as long as the app is not deleted. The username entered will be used as default family photo sharing folder name in the photo slide show / sharing module (Of course user can change that by clicking the change bucket

in the settings button. It will then validate with database and allow the action of change family share folder). The purpose of the pin, meanwhile, is to make it easier for user to re-login once he/she has logged out. After the pin is set, the app check to see if the medication record is set. After the medication record is set, when user opens the app, the main activity will be presented.

The classes in the events package serve as a way to notify the system of when certain events take have taken place. For example, the SendSympDFEvent indicates that we are done trying to send medication data to DreamFactory. All the other events function in the same way.

The main activity is composed of the following four parts:

MainActivity Components breakdown:

Symptom Data Module (upper right block)

1. When a user clicks on the upper right block, the SymptomsActivity.java class is called and the screen below will open. This activity has a JobManager which invokes the classes in the jobs package that eventually sent information to the DreamFactory database. This class contains the methods which facilitate the transfer of symptom data between the UI and the database. For example, the getSymptomsData method takes the information entered into the radio buttons in the upper right block, creates a SymptomDataModel object and then invokes its methods to put the data into a JSON object and eventually send the JSON object into the DF database. Events are sent from these activities when a request is finished sending or received.

Medication Data Module (upper left block)

1. This upper left block deals with the medications that a user is taking. The MedicationActivity has a method which fills in a list of which medication items the user takes. Each one of these items has the medication's name and a radio group that is filled in with the default number of medications the user is supposed to take during their current time interval. MedicationActivity.java sets up the block with a sligtly different UI based on the time of day (morning, afternoon, evening). It also has methods which prepare the medication data to be send to the DreamFactory database by packaging the entered data into JSON objects.

Data Visualization Module (lower left block)

1. When user clicks on the lower left block, LogActivity.java is invoked. It will automatically pull recent records from the database and display it in a sorted list based on the alphabetical order and date.

2. On the upper right of the DataVisualization page, there is a button that leads to a calendar view, which is coded in WebCalendarVisualizationActivity.java. This activity renders the webpage that is used to graphically display the condition of the patients via various means JHeatMap, for example. The part of code that's responsible for website is not included in the MS101 packge.

PhotoSlideShow Module (lower right block)

1. SlideShowActivity is the Activity class for the slideshow, dealing with UI changes. The consistent updates on time and date is run on a background thread in the class, but each update will be sent to the UI thread for changes.
2. WeatherFragment is a fragment contained in the view of slide show. Once initiated, it will download the most updated weather data from the OpenWeatherMap provider in the back end thread, and sent to UI through intents.
3. S3PhotoIntentService class provides back-end photo managing services for the slideshow. It retrieves photos whenever user opens the view and/or manually clicks the update button in the menu. For every photo it has, it will send it to Activity class through intents. It also has a photo caching mechanism (a concurrent HashMap data structure) so that memory taken by photos is under control.

   It is also the class that handles the Amazon COGNITO connection. When S3Client instance is constructed, it will takes in the information provided by user -- account ID, account Hash, authorized and unauthorized policies, and region -- to get connected to the S3 elastic storage.

   Upon request to download (which is handled by broadcast listener in the S3PhotoIntentService class), this backend service pulls photo from the remote storage in the format of bitmap, and stores that into ConcurrentHashMap. This map keeps growing as the worker thread continues to put bitmaps representation of the photos into the map, and stops when the thread finishes downloading the bitmap from the remote storage.

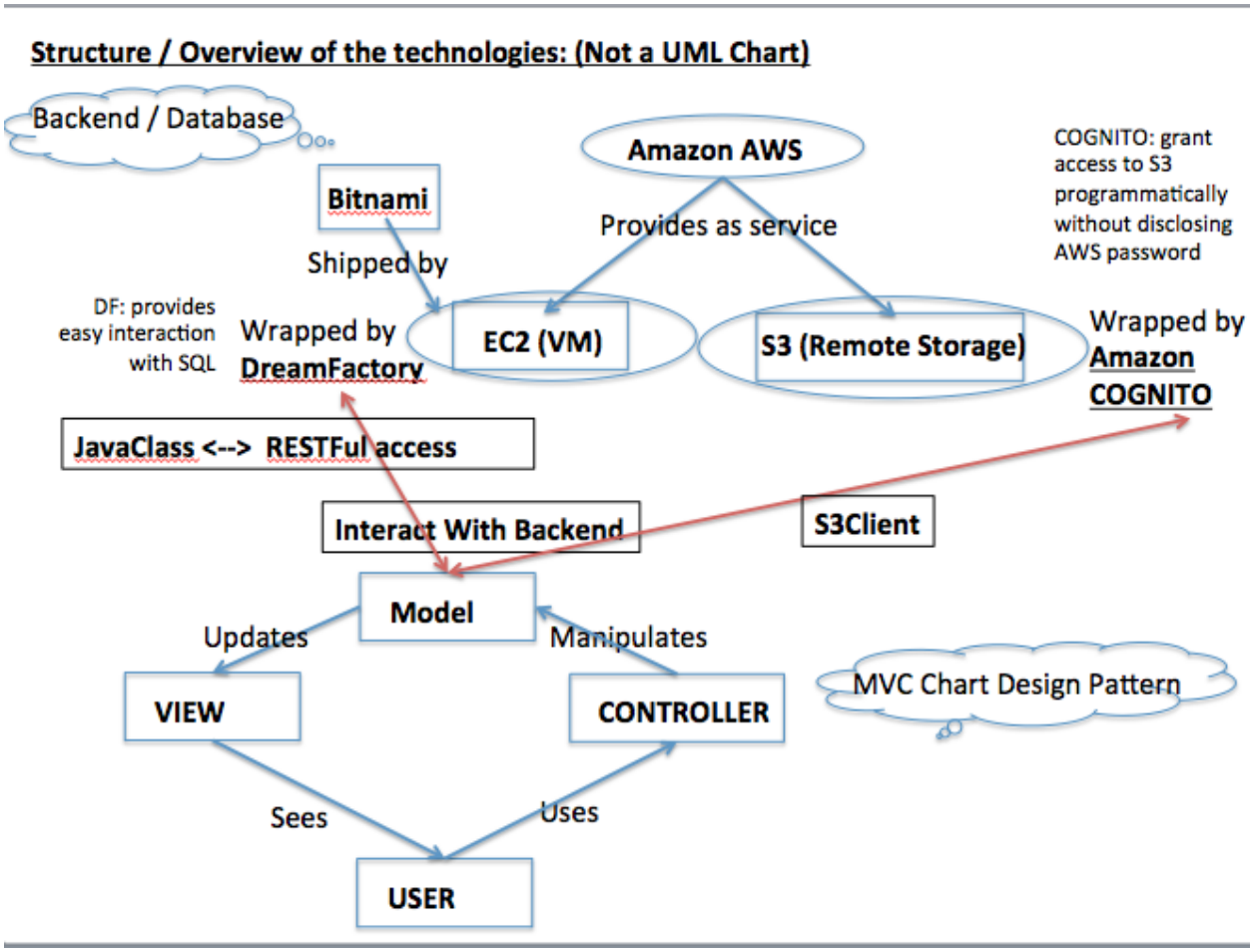
Server Side Notification Update

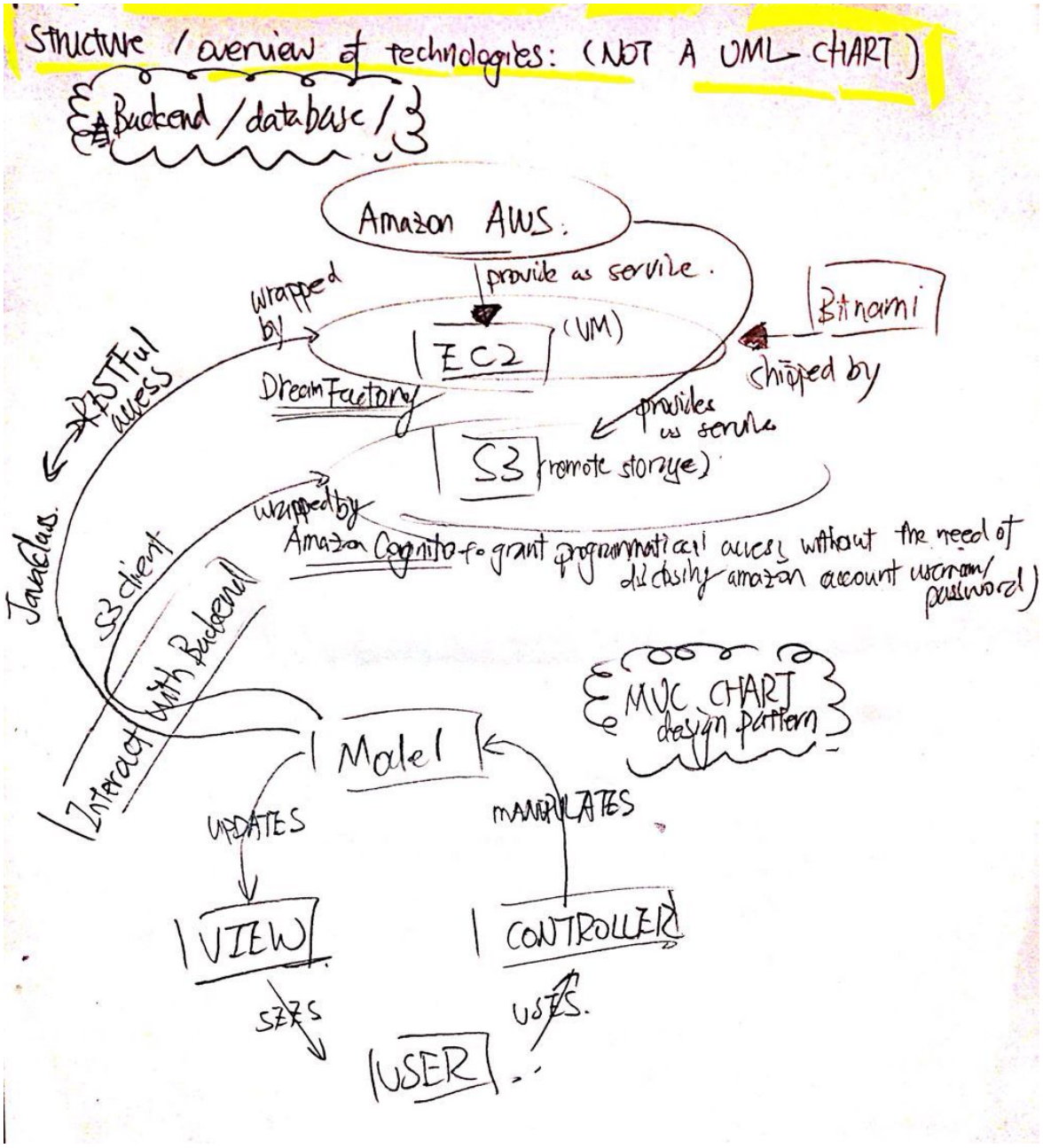
The GCMPushBranch of this respository contains a version of the code where all of the code infrastructure necessary to successfully send notifications from the server has been written. Go to the GCMPushBranch, select the docs folder and view the document called PushNotifications.pdf for complete instructions on the status of this feature. Additionally, this document contains instructions on what is necessary to fully integrate this feature.

Technologies Our Code Depends On

1. **Dreamfactory API** that wraps around MySQL server which provides easier Database access.
2. **Amazon S3 (Elastic Storage)** that provides storage spaces for photo-sharing service
3. **Amazon COGNITO (Token Generating Machine)** that provides token for program to access to **S3** without exposing username and password of the Amazon account.
4. **The OpenWeatherMap APIs** for weather data retrieval

Graphical Representation of Overall Structure of Code, and Technologies Our Code Depends On





Dreamfactory Database Schema
Refer to the tables in db_tables.txt

Classes in the app used to connect send and retrieve information between the UI/Database

1. Backend.java has methods such as getFromDF and sendToDF that are used within the application to interact with the database. The DF url is hardcoded into this class.
2. All of the UI classes except for the photoslideshow classes metioned previously use these methods to interact with the database (ie. Medication Activity, Symptoms Activity).