The University of British Columbia

L o c a l l y

Design Document

Angy Chung

Anna Gudimova

David Jung

Mo Kiani

Andy Lin

Alena Safina

October 17, 2016

INTRODUCTION

Locally is an Android application that will allow users to view locally grown produce being sold at farmer's markets, check what's in season, and view nearby vendors. The data will come directly from the vendors themselves as they can update their daily produce stock through our application.

This documents explains architecture and design of the system to be built.

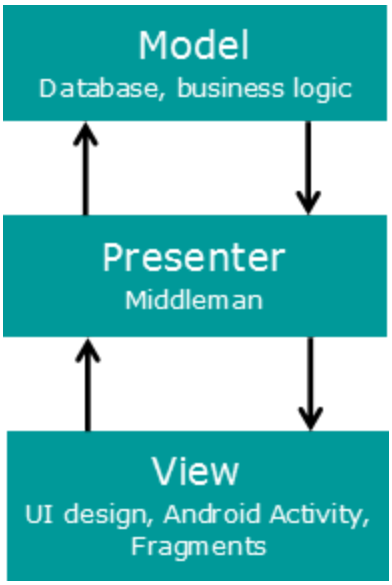
SYSTEM ARCHITECTURE

The application consists of the following major components:

1. Non relational database (specifically AWS DynamoDB) that will store tables with application data with different access permissions for consumers and vendors.
2. Java for the Android application backend to handle user interface and requests to the database
3. XML for the frontend Android application UI design following Google's Material Design guidelines
4. MVP design pattern

Model-View-Presenter Architecture

The model-view-presenter (MVP) architecture pattern is a pattern adapted from the of the model-view-controller (MVC) pattern. The components of the MVP pattern are:

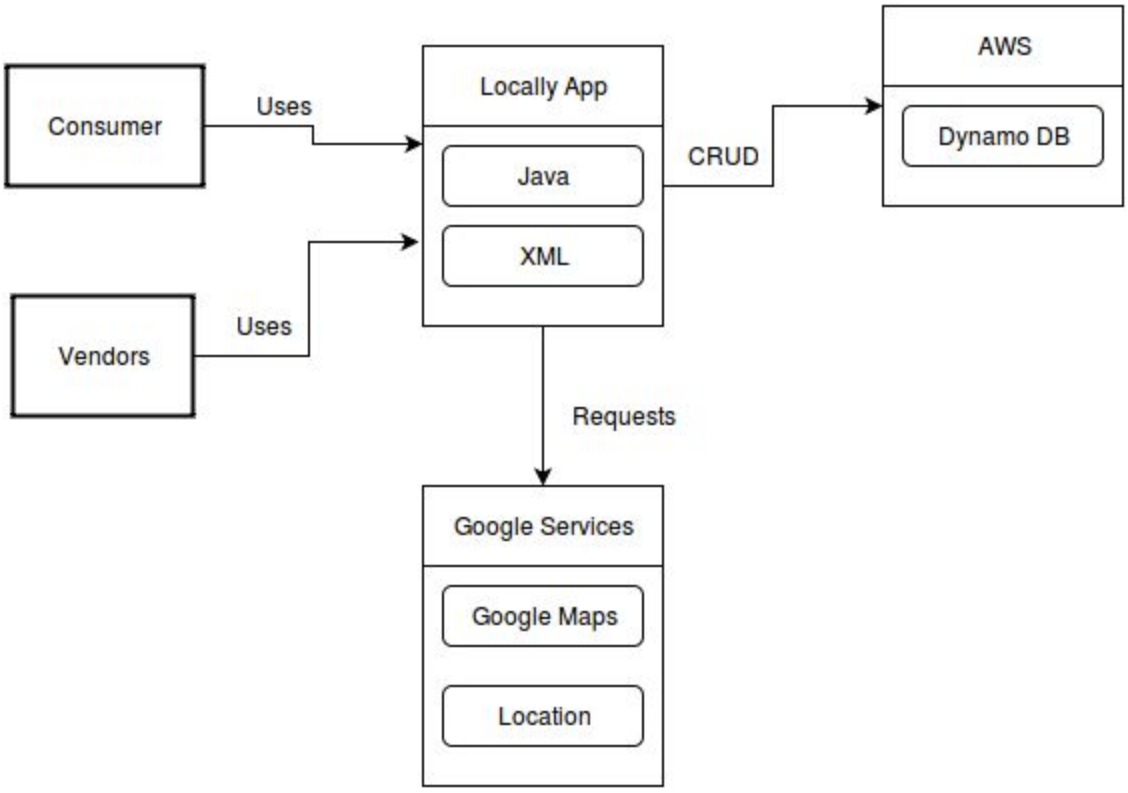


Model - The model is responsible for the business logic of the application which defines how data can be created, deleted, modified, and stored. On Android it is a data access layer, a AWS DynamoDB database in our case.

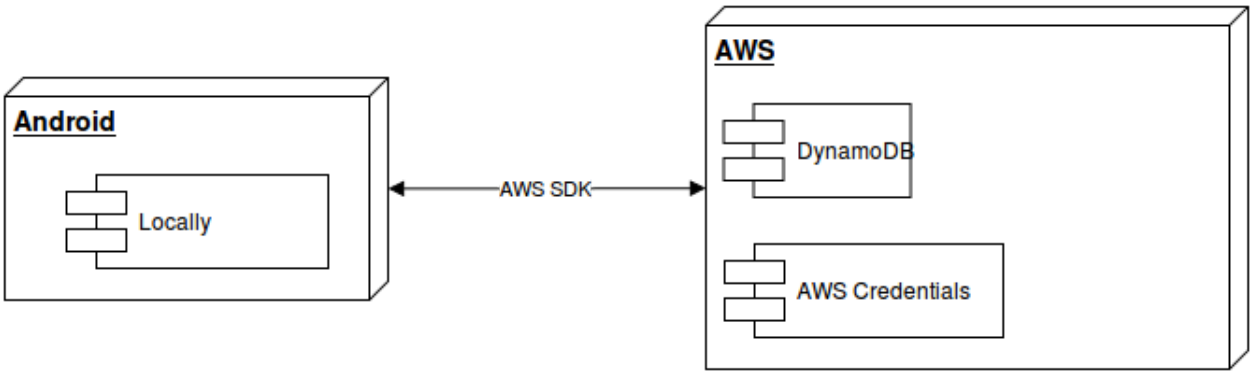
View - The view is a predominantly passive element that displays visual data, reacts to user actions, receives user input (events) and forwards it to the presenter. On Android, this could be an Activity, a Fragment, an android.view.View or a Dialog.

Presenter - The presenter serves as the middleman between the model and the view by receiving user events from the view, retrieving data from the model, and formatting data for display to the view. It is a codelayer dealing with all the background processing on Android.
The MVP model differs from the more traditional MVC model as the latter's View component can be more active and retrieve data directly from the Model. Thus, we chose to employ the MVP model for our project as it affords a greater separation of concerns between the three core architectural layers and allows for easier unit testing of the project.

Dynamic View



Deployment Diagram



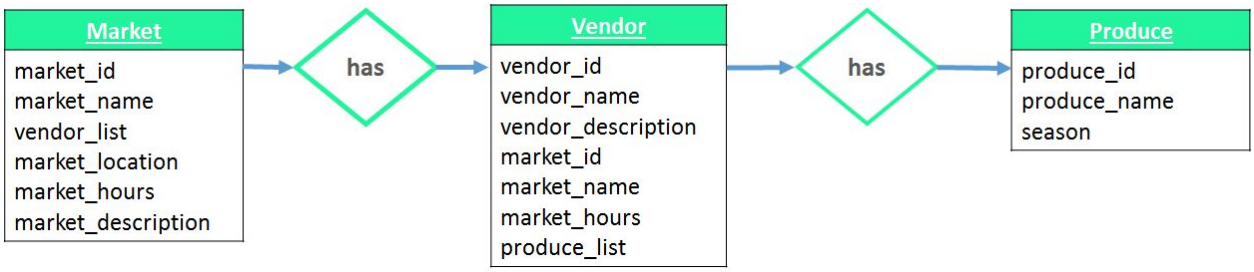
DATABASE DESIGN

DynamoDB

AWS offers an Android SDK for a DynamoDB allowing for CRUD (Creation, Reading, Updating and Deleting) database entries. By choosing DynamoDB we do not have to worry about load balancing, scaling, handling credentials, among other administrative tasks. Other potential solutions included Firebase or using REST but with support being excellent for DynamoDB along with the developers having more experience using AWS, DynamoDB was the natural decision.

Database Diagram

As of now the pseudo (NoSQL) Entity-Relationship diagram is as follows



GUI

As mentioned above, the user interface will be built using XML to design application views as per Google's Material Design guidelines. We will describe the design for main views below.

Home Page

We will place the navigation Sliding Sidebar (Hamburger) Menu with Navigation Drawer Icon in the top left corner of the home page. The menu shall contain main destinations for our app, such as map, settings, log in as vendor, etc. See the prototype on the image 1.

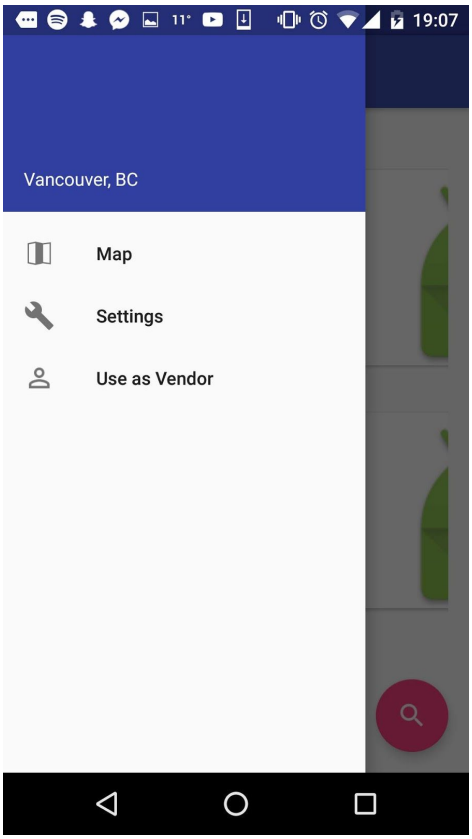


Image 1 - *Hamburger Menu*

Below the menu we will have another elements:

1. Search bar - for users to search produce, vendors, markets (by name)
2. Sliding horizontal bar of markets' logos for user to scroll and get acquainted with existing markets.
3. Nearby, In Season, Calendar buttons in row at the bottom of the page linked to Map View, list of produce in season, and Calendar View described below. See the image 2 mockup for reference.

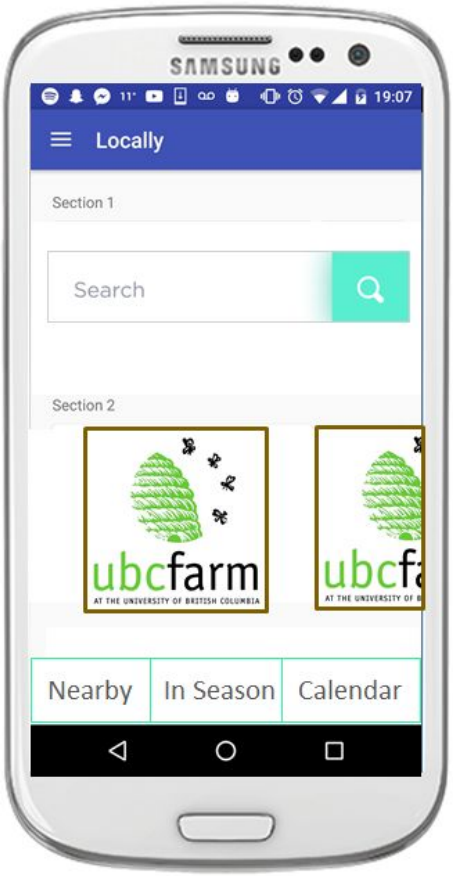


Image 2 - *Home Page Mockup*

Calendar View

Calendar view button leads the user to the page displaying the list of market names, locations and hours/months of operation. Once the user clicks on the market name they shall be directed to the page containing broader market description and directions. See the prototype design for the calendar view on image 3.

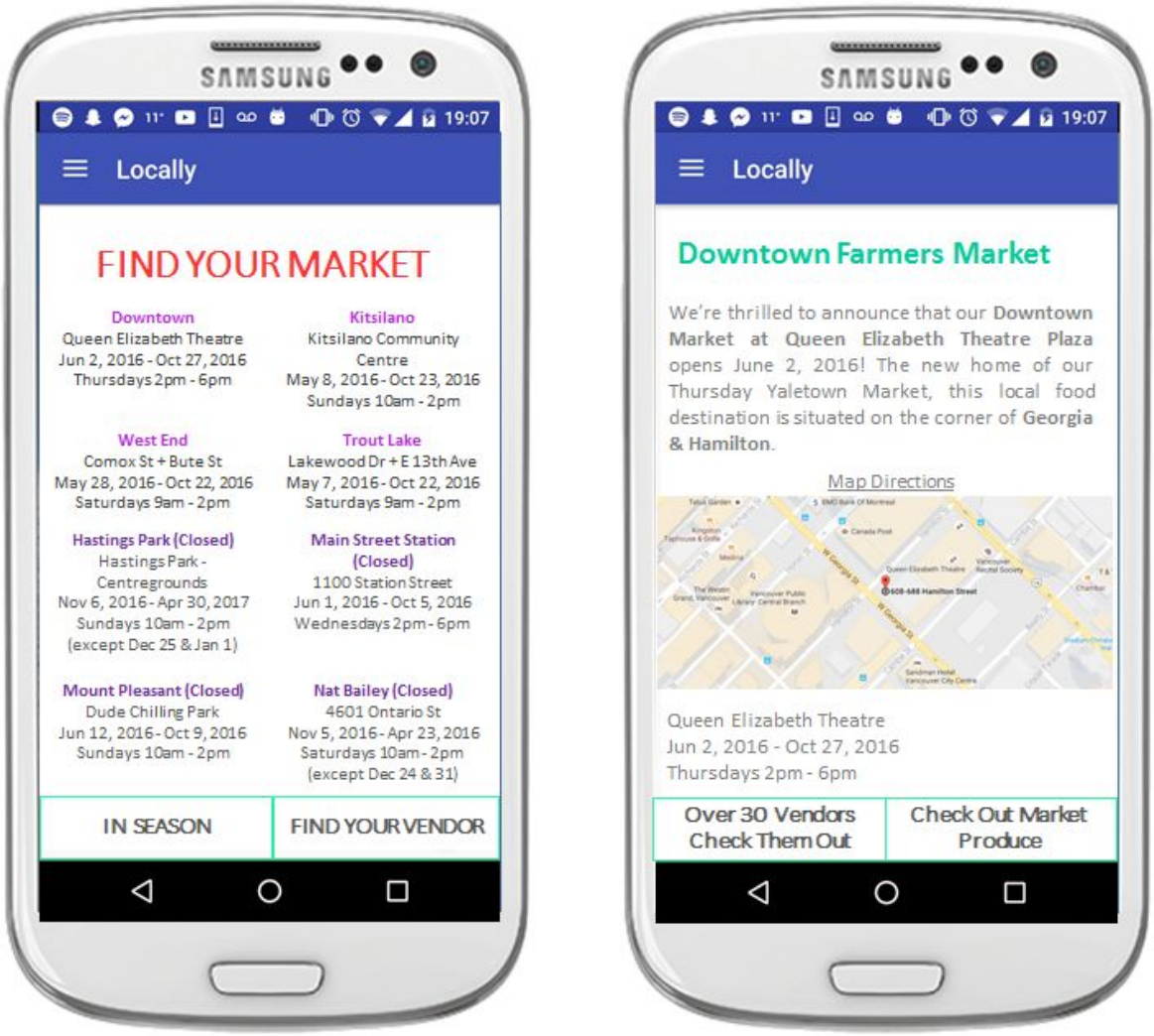


Image 3 - *Prototype for Calendar View*

Map View

The map view presents a visual way to discover markets and vendors around a user. It utilizes the Google Maps API to display the locations of vendors as flags, which are linked to their respective vendor detail pages for users to learn more about a specific vendor.

Vendor Registration/Log In

Once the user has selected the "Use as Vendor" button in the navigation drawer (hamburger), they will be brought to the vendor registration/log in page where they will be able to log in with

their user credentials into our system to update their stock, or go through our registration process to register themselves as a vendor in our system.

Vendor List View

This view will display a list of vendors, with details such as vendor name, vendor location, the consumer's distance away from the vendor and whether the vendor is currently open or closed. If the user wishes to learn more about any vendor within the vendor list, they can double click a vendor list item to open up the vendor detail view. Within, this view, users will be able to view all of the details mentioned above, as well as additional vendor descriptions, their hours, and a produce list that describes their in season products.

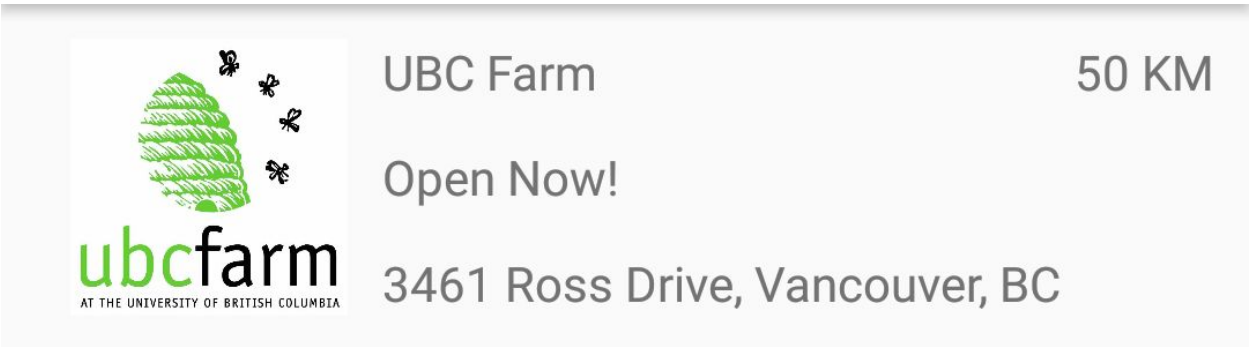


Image 4 - *Vendor List Item Mockup*

These were the major views that we will use in our design. Other views, like list of search results or list of produce in season will be used in our application as well.