Description
Intended User
Features
User Interface Mocks
Screen 1
Screen 2
Screen 3
Screen 4
Key Considerations
How will your app handle data persistence?
Describe any corner cases in the UX.
Describe any libraries you'll be using and share your reasoning for including them.
Describe how you will implement Google Play Services.
Next Steps: Required Tasks
Task 1: Project Setup
Task 2: Implement UI for Each Activity and Fragment
Task 3: Your Next Task
Task 4: Your Next Task
Task 5: Your Next Task

GitHub Username: jruetas

Abell Route Planner

Description

Abell Route Planner will help plan an optimal route to travel to multiple locations for Abell Pest Control service technicians. The goal is to allow entry up to 100 locations and to use the Travelling Salesman Problem (TSP) algorithm to order the locations in such a way as to provide the shortest distance of a path that includes all locations a technician wants to visit. The initial list given will be provided by connecting to a web service and downloading service tickets for a particular day for a specific technician. A technician can then select which locations they would like to visit with the ability to add new locations. After a technician selects the locations they can use a function to create a route. Once a route is created, the technician can view the map using Google Map related APIs.
The problem this solves is providing a route a technician can follow to complete schedule service for a particular day. Once a map is created the technician can use Google Map components to do such things as getting directions to a customer location, help them plan their day, etc.

Intended User

The intended user for this app is only Abell Pest Control Service Technicians.

Features

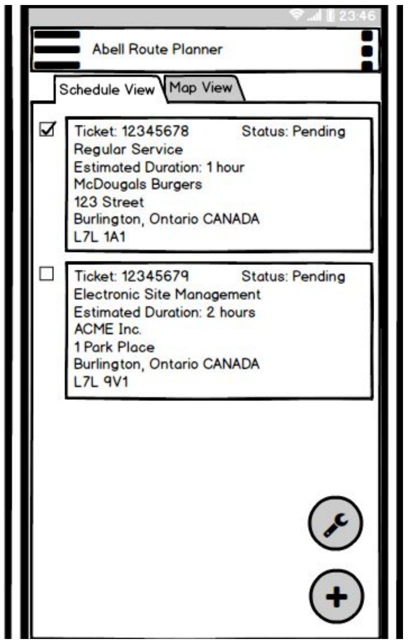
The main features of this app:

- Downloads service tickets and Customer information from a web service
- Saves downloaded information to a local database
- Ability to view service ticket and Customer information
- Ability to add service ticket and Customer information
- Ability to edit service ticket and Customer information
- Ability to delete service ticket and Customer information
- Can select/deselect locations for routing
- Plots the shortest distance that includes all selected locations using the Travelling Salesman Problem (TSP) algorithm
- Using Google Maps related APIs can do some of the following:
    - Get driving directions to a location
    - Ability to have an estimate of arrival/departure times to the locations

User Interface Mocks

The following User Interface Mocks were created using Balsamiq.

Screen 1



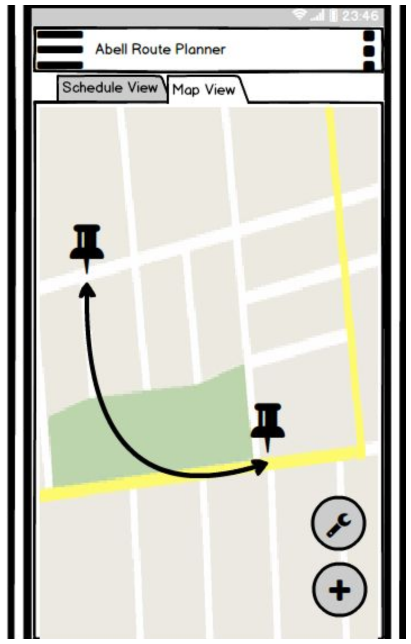
☑ Ticket: 12345678 Status: Pending
Regular Service
Estimated Duration: 1 hour
McDougals Burgers
123 Street
Burlington, Ontario CANADA
L7L 1A1

☐ Ticket: 12345679 Status: Pending
Electronic Site Management
Estimated Duration: 2 hours
ACME Inc.
1 Park Place
Burlington, Ontario CANADA
L7L 9V1

The first screen (tab) is the Schedule View. After a technician downloads the service tickets he has to work on for the day he can perform any of the following:

- Select which service tickets he will work on for the day by either checking or unchecking
- Add, edit or delete service tickets:
    - Adding is performed by clicking the + symbol on the bottom right.
    - Editing a service ticket is done by clicking on the service ticket information to bring up the edit screen.
    - Deleting a service ticket is done by swiping left. A prompt will ask to verify if this is the desired action. If the technician chooses ‘Yes’ only then will the service ticket be deleted.
- Optimize the route action is done after the technician select the wrench icon on the bottom right. The problem will then try to figure out the optimal route using the selected service tickets.

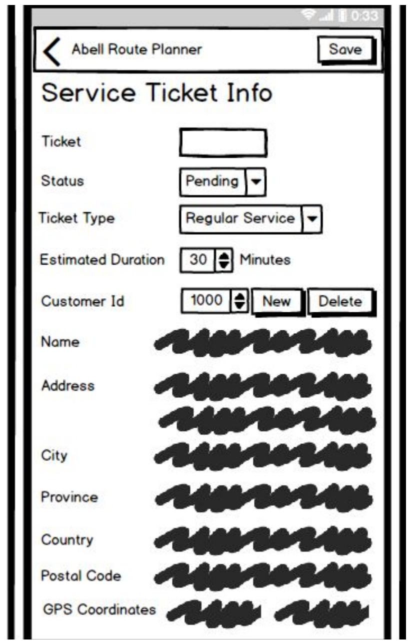
Screen 2



The second screen tab is the map view. Here the technician can view the service tickets he/she selected on the schedule view. If the technician pressed the Optimize button (wrench) while on the schedule view the optimal route will be already plotted otherwise they can select it while on this screen. While on this screen the technician can perform the following tasks:

- Add a service ticket
- Optimize the route for the selected service tickets
- Obtain driving directions to a service ticket location by pressing on the location pin and then selecting driving directions.

Screen 3



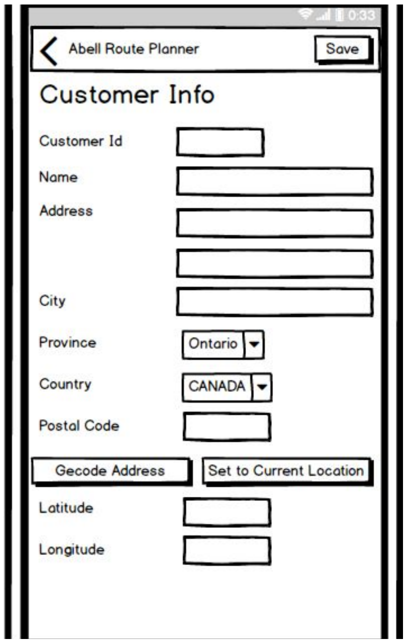
Service Ticket Info

Ticket	
Status	Pending
Ticket Type	Regular Service
Estimated Duration	30 Minutes
Customer Id	1000 New Delete
Name	
Address	
City	
Province	
Country	
Postal Code	
GPS Coordinates	

If the technician has chosen to edit or create a new service ticket he/she will be presented with the following screen. On the screen they can enter the following:

- Ticket
- Status
- Ticket Type
- Estimated Duration
- Select a Customer
- Create (New) or Delete a Customer
    - If they selected to create a Customer they will be presented the Customer Info screen.

Screen 4



If the Service Technician has selected to either create or edit a Customer they will be presented with the following screen. On the screen they can enter the following information:

- Customer Id
- Name
- Address / Address2
- City
- Province/State
- Country
- Postal Code/Zip Code
- Latitude
- Longitude

The Longitude and Latitude can be entered by the following:

- Manually
- Geocoding the entered address
- Set to the GPS coordinates of the current location

Key Considerations

How will your app handle data persistence?

I intend to build a Content Provider which communicates to a local SQLite database to handle data persistence.

Describe any corner cases in the UX.

To make the project accomplishable I intend to create one main screen with 2 tabs which includes one for a schedule list and the other for a map. There will be secondary screen for viewing and editing information.

Describe any libraries you'll be using and share your reasoning for including them.

Initially I plan on including the OkHttp library as the Http client. This will allow the app to communicate to the web service and request service ticket and customer information.

Describe how you will implement Google Play Services.

I plan on using the Google Maps APIs for any map related functionality When adding or editing customer information I intend on using the Google Maps Geocoding API to provide editable GPS coordinates. After an initial route has been created I plan on using Google Maps Directions API and the Google Maps Matrix API to provide directions from one location to another and to calculate and estimated times of arrival.

Next Steps: Required Tasks

Task 1: Project Setup

The steps to setup and/or configure this project include:

- Create a new project and name it Abell Route Planner. The default production and debug versions are adequate.
- Add all the above listed libraries to the build.gradle file
- Create all the required Sync Adapter components:
  - Authenticator
  - Content Provider
    - Database Contract
    - Database Class that extends the SQLiteOpenHelper
    - Content Provider
  - Sync Adapter

Task 2: Implement UI for Each Activity and Fragment

Build the following:

- Build UI for MainActivity
- Build UI for Schedule View Fragment
- Build UI for the Map View Fragment
- Build UI for the Service Ticket Info Activity
- Build UI for the Customer Info Activity
- Build UI for the Create Account Activity. I did not include it in the above screen mocks since this will be a simple screen which only includes Username and Pin (which is our version of a password and is composed of 4 to 6 digits)

Task 3: Create the required Sync Adapter Classes

Create the necessary Sync Adapter Classes and verify that a username/pin combination gets validated to a web service.

Task 4: Create all the UI's

Create all the UI's and layouts listed above.

Task 5: Link the data to the UI

Link the data retrieved from the Sync Adapter to the UI's created in step 4.