Cicely Lambright, Matthew McClellan, Nolan Reed, Chi Zhang

Team 5 Lab 2 Reflection Document

**Table of Contents**

Roles of Team Members…………………………………………………………….……..…..2

Status of Features…………………………………………………………………………..…..3

Team Member Communication Strategies……………………………………………….…..7

Experience Gained…………………………………………………………………………..….8

Top 5 Risks Moving Forward…………………………………………………………………..9

**Roles of Team Members**

Matthew assumed the role of coordinator and helped to assign responsibilities to other team members. Additionally, Matthew took primary responsibility for implementing feature six.

Cicely was responsible for ensuring the quality of Matthew’s feature six implementation through unit testing and bug fixing. Cicely made use of the unit tests drafted by Chi and adapted them further to specifically fit our implementation.

Chi worked closely with Nolan to refactor the code. Chi drafted the unit tests for Cicely to use. The identification of code smells and code refactoring was completely performed by the Chi and Nolan.

As stated above, Nolan was heavily involved with the refactoring of our previous code.

**Status of Features**

Feature M-1

Fully-implemented

As requested, a dialogue is presented where the user is prompted to enter an address. This dialogue is created in its own Android fragment. On pressing the search button, when the search button is pressed, the user input is passed to the second feature.

Feature M-2

Fully-implemented

Given user input from the first feature, this feature uses a Mapbox GeoCoder object to retrieve a latitude and longitude at which to place the marker on the mapview. This mapview and its associated markers are created within a separate fragment than M-1.

Feature M-3

Fully-implemented

On clicking a marker from feature two, an information window is created above the marker, which displays information about the address at which the marker is placed. The information is determined by sending the coordinates from the GeoCoder back through to get the full list of information about the address, rather than just the user’s entered string. When the marker is pressed again, the window disappears..

Feature M-4

Full-implemented

The text “navigate here” was added to the Information window by creating a custom information window. The portion of the window which holds the “navigate here” text was equipped with an onClickListener, which allows it to detect when it is clicked, and to subsequently perform its functionality. This method was used as a workaround to prevent the team from actually having to implement javascript and other web-related functionality into the code, which would only hurt the performance of the application. This method, rather, cleanly performs the exact same task, while still emulating the feel of a hyperlink.

Once clicked, the “navigate here” listener determines the user’s current location, which is actually hard-coded into the application since the computer does not give the emulator the ability to determine the exact location it is being run from, and sends the current location along with the searched location off to Mapbox in a routes query. This is done asynchronously within the Android code because Android does not support web requests being performed on its main thread. Once the route data is retrieved, it is parsed by being passed through a series of objects, and the route is eventually overlaid on the mapview between the two markers.

Feature M-5

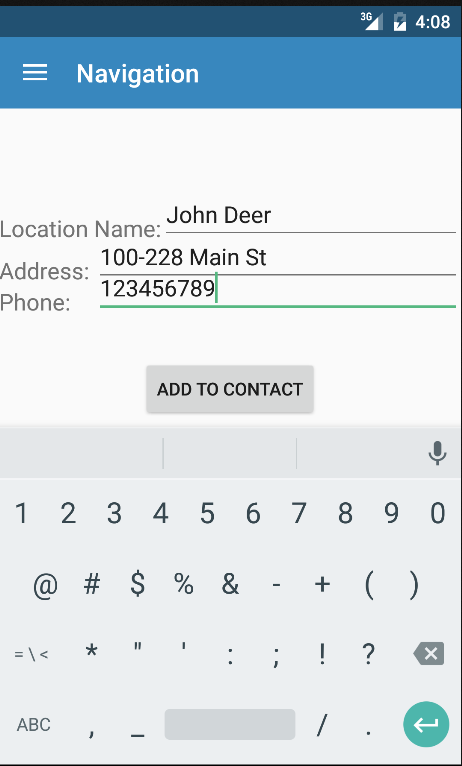
Fully-implemented

If two or more routes are found during feature M-4’s route query, a new dialogue is created via a separate fragment which allows the user to choose between the desired routes using the select button next to the desired route. This data all existed within the fourth feature, but a list adapter was used to allow the dialogue to be dynamically created based on the number of routes to be displayed.

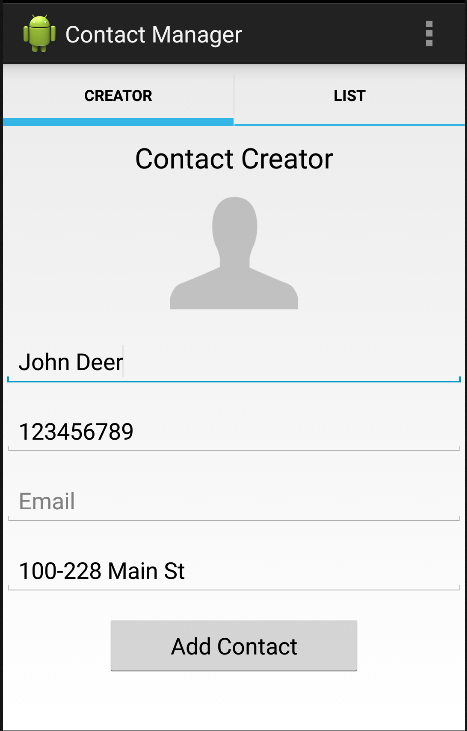
Feature M-6

Fully-implemented

When the info window created by feature is long pressed (long press was used instead of double tap because the android documentation explicitly stated that long press was better), the dialogue opens a new fragment which displays the address form the info window along with two lines for phone and location name which the user may populate.



When “Add to Contact” is pressed on this screen, Contact Manager is opened, and relevant fields are populated.



Feature C-1

Fully-implemented

When the settings button was clicked, the privacy dialogue initialized. When the address toggle is toggled, and contact’s address becomes a “hyperlink” just like the hyperlinks seen in feature M-4. We used two separate activities switch from the create contact dialogue into the privacy dialogue. When the privacy dialogue is left, the state of the dialogue will not be saved for future uses.

**Team Member Communication Strategies**

During Lab two, we strove to greatly improve our team communication. Each lab session, we would beet as a group and offer progress reports in person. During lab times, Matthew would check in with each group member and make sure that each member was on the same page as everyone else. Matthew and Cicely remained in contact during the week through text message to ensure that feature development was moving along smoothly.

**Experience Gained**

As a group, we learned more about the different methods of refactoring by applying them throughout our code. We also learned how to successfully send information from one application to another. We found that allowing the highly independent members of our team to perform their assigned tasks on their own with frequent “checkpoint” conversations being held to ensure synchronization worked well for our group.

One thing that did not work well included our bottlenecking of the refactoring-focused members of the team by failing to properly use our source control and provide up-to-date code to those members for a few days. This issue was entirely a mistake, and was resolved after a relatively short time by good team communication. Otherwise, our team did not really have to suffer from any team-related engineering failures.

**Top 5 Risks Moving Forward**

1. As features become more technically involved, it may be a difficult to keep the team’s technical proficiency progressing at the same rate as the feature involvement.
2. Making sure that each feature provides a robust user experience involves a great deal of critical thinking and creative usage of the features. Testing these features properly can be difficult.
3. Making sure that our deliverables promised to the group at large are delivered as expected so that the workflow of the rest of the group is not bottlenecked
4. Ensuring that our code is well documented is highly important, as future features will take longer to implement if our code is not documented internally.
5. Maintaining a high level of contact and coordination will require us as individuals to continue our hard work in this area moving into lab three.

Since the last lab, the risks associated with the extreme difficulty of the features to be implemented were mitigated by Dr. Niu and Wentao decreasing the difficulty of Lab 2. This relieved a tremendous amount of pressure and risk from Lab 2. Besides this change, we, as a team, were better about communicating during class time and were able to divide the labor much more evenly this time around. This helped reduce the risk of miscommunication and bad progress reporting that we feared for Lab 2.